BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 01-126736

(43)Date of publication of application : 18.05.1989

(51)Int.Cl. G06F 9/46
G06F 9/44

(21)Application number : 63-207998

(22)Date of filing : 22.08.1988

(71)Applicant : WANG LAB INC

(72)Inventor : KHOYI DANA
SOUCIE MARC S
SURPRENANT CAROLYN E
STERN LAURA O
PHAM LY-HUONG C

(30)Priority

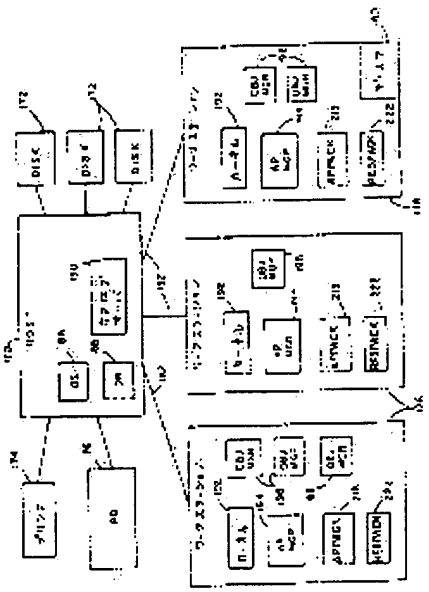
Priority number : 87 88622 Priority date : 21.08.1987 Priority country : US

(54) DATA PROCESSING SYSTEM

(57)Abstract:

PURPOSE: To share data resources among various programs by providing an open type frame work in a customizable computer system.

CONSTITUTION: An RESPACK 222 is the pack of services and functions used for accessing resources and correcting them. Some program or user is used, the device of the RESPACK 222 is used and the resources are prepared. Or existing resources are copied and the copied resources are corrected so as to match with program function and user specified requirements. In these cases, the customized resources are specified for the program, function, and user prepared and corrected for them.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　平1-126736

⑤1Int.Cl.⁴		識別記号	庁内整理番号	㊸公開　平成1年(1989)5月18日
G 06 F	9/46	340	B-7056-5B	
	9/44	330	Z-8724-5B	
	9/46	340	F-7056-5B	審査請求　未請求　請求項の数 27 (全87頁)

㊹発明の名称　データ処理システム

㉑特　願　昭63-207998

㉒出　願　昭63(1988)8月22日

優先権主張　㉜1987年8月21日㉝米国(US)㉛886622

㉒発明者　ダナ・コーイ　アメリカ合衆国マサチューセッツ州01826, ドラカット, トウラウト・プルツク・ロード　207

㉒発明者　マーク・サン・ソウシー　アメリカ合衆国マサチューセッツ州01879, タインスボロ, シャーバーン・アベニュー　33, ナンバー　8

㉑出願人　ウオング・ラボラトリーズ・インコーポレーテッド　アメリカ合衆国マサチューセッツ州01851, ローウエル, ワン・インダストリアル・アベニュー　(番地なし)

㉔代理人　弁理士　湯浅　恭三　外4名

最終頁に続く

明細書の浄書(内容に変更なし)

明　細　書

1.〔発明の名称〕

データ処理システム

2.〔特許請求の範囲〕

1)　データがタイプされたオブジェクトとして表示されるデータ処理システムにおいて、

(A)　少なくとも1タイプのオブジェクトに対して少なくとも1オペレーションを各々が実行するようにされた複数のオブジェクトマネージャと、

(B)

(C)　リクエストしているオブジェクトマネージャからオブジェクトの識別とオペレーションの識別とを受け取り、

(D)　識別されたオブジェクトのタイプのオブジェクトに対して識別されたオペレーションを実行できるオブジェクトマネージャを識別し、および

(E)　識別されたオブジェクトマネージャを呼び出し、かつ識別されたオブジェクトマネージャに対して識別されたオブジェクトに識別されたオペレーションを実行するようとのリクエストを通信するマネジメント手段

とを含むことを特徴とするデータ処理システム。

2)(A)　タイプされたオブジェクトを表示するデータを記憶する手段と、

(B)　各々が少なくとも1タイプのオブジェクトに対して少なくとも1オペレーションを実行するようにされた複数のオブジェクトマネージャと、

(C)　それに対して、いずれのオブジェクトマネージャもオブジェクトおよびオペレーションを識別できるマネジメント手段であって、その識別に応答して適当なオブジェクトマネージャを呼び出してオブジェクトに対してオペレーションを実行させるマネジメント手段とを

含むことを特徴とするデータ処理システム。

3)　各々が少なくとも1タイプのオブジェクトに対して少なくとも1オペレーションを実行するようにされた複数のオブジェクトマネージャを

有するオブジェクトベースのデータ処理システムに　いて、アプリケーションマネージャが、

（A）　リクエストしているオブジェクトマネージャからオブジェクトの識別とオペレーションの識別とを受け取る手段と、

（B）　識別されたオブジェクトのタイプのオブジェクトに識別されたオペレーションを実行できるオブジェクトマネージャを識別する手段と、

（C）　識別されたオブジェクトマネージャを呼び出し、識別されたオブジェクトマネージャに、識別されたオブジェクトに識別されたオペレーションを実行するようにとのリクエストを通信する手段とを

含むことを特徴とするデータ処理システム。

4）　請求項1に記載のシステムにおいて、複数のオブジェクトマネージャが、各タイプのオブジェクトに対して1組の標準オペレーションの各々を実行できるオブジェクトマネージャを含むことを特徴とするデータ処理システム。

5）　請求項2に記載のシステムにおいて、複数

(2)

のオブジェクトマネージャが各タイプのオブジェクトに対して一組の標準オペレーションの各々を実行できるオブジェクトマネージャを含むことを特徴とするデータ処理システム。

6）　請求項1に記載のシステムにおいて、各タイプのオブジェクトに対して省略時のオペレーションが定義されており、かつ複数のオブジェクトマネージャが各タイプのオブジェクトに対して対応する省略時オペレーションを実行できる少なくとも1個のオブジェクトマネージャを含むことを特徴とするデータ処理システム。

7）　請求項2に記載のシステムにおいて、各タイプのオブジェクトに対して省略時オペレーションが定義されており、複数のオブジェクトマネージャが各タイプのオブジェクトに対して対応する省略時オペレーションを実行できる少なくとも1個のオブジェクトマネージャを含むことを特徴とするデータ処理システム。

8）　請求項1に記載のシステムにおいて

（i）　マネジメント手段がマネージャプロセ

スからなり、

（ii）　マネジメント手段による受け取りが、マネージャプロセスによる、リクエストしているオブジェクトマネージャからのプロセス間通信の受け取りを含み、および

（iii）　マネジメント手段による呼び出しが結果的にマネージャプロセスがオブジェクトマネージャプロセスを生成させるようにする

含むことを特徴とするデータ処理システム。

9）　請求項1に記載のシステムにおいて、マネジメント手段がオブジェクトをオブジェクトタイプに関連させるデータベースをアクセスする手段を含むことを特徴とするデータ処理システム。

10）　請求項1に記載のシステムにおいて、マネジメント手段がオブジェクトタイプをオブジェクトマネージャに関連させるデータベースをアクセスする手段を含むことを特徴とするデータ処理システム。

11）　請求項1に記載のシステムにおいて、リクエストしているオブジェクトマネージャにより提供されたオブジェクトの識別がリンク識別子からなることを特徴とするデータ処理システム。

12）　タイプされたオブジェクトとしてデータが表示されるデータ処理システムにおいて、

（A）　少なくとも1タイプのオブジェクトに対して少なくとも1オペレーションを各々実行するようにされた複数のオブジェクトマネージャと、

（B）(C）　オブジェクトをオブジェクトタイプと関連させ、

（B）　リンク識別子を特定のオブジェクトと関連させ、

および(C）　オブジェクトタイプをオブジェクトマネージャと関連させる

システムデータベースと、

（A）(B）　リクエストしているオブジェクトマネージャからリンク識別子とオペレーションの識別とを受け取り、

（C）　システムデータベースにアクセスしてリンク識別子が参照しているオブジェクトを識別し、

(D) システムデータベースにアクセスして識別されたオブジェクトのタイプを検出し、

(E) システムデータベースにアクセスして、識別されたオブジェクトのタイプのオブジェクトに識別されたオペレーションを実行できるオブジェクトマネージャを識別し、および

(F) 識別されたオブジェクトマネージャを呼び出し、識別されたオブジェクトマネージャに、識別されたオブジェクトに識別されたオペレーションを実行するようリクエストを通信するマネジメント手段、

とを含むことを特徴とするデータ処理システム。

13) リンクされたオブジェクトを含むデータ処理システムにおいて、

(A) 複数のオブジェクトマネージャと、

(B) 複数のオブジェクトと、

(C) 各リンクと関連したオブジェクトを識別する情報を記憶する記憶手段と、

(D) 記憶されたリンク情報にアクセスする手段であって、複数のオブジェクトマネージャの

(3) 全てに対してリンク情報を提供しうるアクセス手段と、

含むことを特徴とするデータ処理システム。

14) 請求項13に記載のシステムにおいて、前記記憶手段が、リンクがデータリンクであるか否かの指示を各リンクに対して含むことを特徴とするデータ処理システム。

15) 請求項13に記載のシステムにおいて、リンクの中の少なくともあるリンクは子供オブジェクトからのデータを親のオブジェクトにリンク結合させ、記憶手段が各リンクに対して、

(i) 子供オブジェクトからリンクされたデータが修正されるとき、

(ii) 親オブジェクトが開放されたとき、あるいは

(iii) 特別の更新リクエストがなされたときのみ、

リンクが更新されるべきことを各更新状態が指示できる更新状態の値を記憶することを

特徴とするデータ処理システム。

16) リンクされたオブジェクトを含むデータ処理システムにおいて、

(A) 複数のオブジェクトマネージャと、

(B) リンクマーカを含む複数の親オブジェクトと、

(C) リンクマーカを含むオブジェクトからは分離しており、子供のオブジェクトをリンクに関係づける情報を記憶する手段と、

(D)(i) リクエストがリンクを識別することによりリンクされたオブジェクトを識別する場合に、リンクされたオブジェクトにオペレーションを実行すべきとのリクエストを受け取り、および

(ii) 記憶されたリンク情報にアクセスしどのオブジェクトがリクエストで識別されたリンクの子供オブジェクトであるか検出するために、各オブジェクトマネージャの各々にとって利用可能のリンクマネジメント手段

とを含むことを特徴とするデータ処理システム。

17) 請求項16に記載のシステムにおいて、リンクマネジメント手段が受け取られたリクエストに応答して識別されたリンク結合ずみオブジェクトにリクエストされたオペレーションを実行するようオブジェクトマネージャを呼び出すことを特徴とするデータ処理システム。

18) 請求項17に記載のシステムにおいて、リンクマネジメント手段が応答するリクエストはリンク更新リクエストであることを特徴とするデータ処理システム。

19) 請求項16に記載のシステムにおいて、各リンクマーカが、該マーカが記憶されているオブジェクト内では一意であるリンク識別子を含み、かつリンクが親オブジェクトおよび対応するリンクマーカのリンク識別子を識別することにより識別されることを特徴とするデータ処理システム。

20) 請求項13に記載のシステムにおいて、リンクされたデータのコピイを記憶する、リンクされたデータのコピイの記　手段をさらに含むことを特徴とするデータ処理システム。

21) 請求項20に記載のシステムにおいて、リ

ンクされたデータのコピイの記憶手段が、別のオブジェクトへのデータリンクを有する各オブジェクトに対応する個別のファイルにおいて記憶することを　徴とするデータ処理システム。

22) ユーザの入力のあるものが第１のプロセスに導かれ、ユーザの入力の他のものが第２のプロセスに導かれるマルチ処理システムにおいて、マッチメーカが、

（A） 複数のプロセスのいずれかから、指示しているプロセスの果すデータ交換の役割の指示を受け取る手段と、

（B） 受け取られた役割の指示を記憶する手段と、

（C） 各々の受け取られた役割の指示が記憶された役割の指示と適合するか否かを検出するマッチ手段と

（D） そのプロセスから受け取った役割の指示が別のプロセスから受け取られた役割の指示と適合した旨をプロセスに通知する手段

とを含むことを特徴とするマルチ処理システム。

23) 請求項22に記載のシステムにおいて、そのデータ交換役割が適合するプロセスの間の通信を設ける手段をさらに含むことを特徴とするマルチ処理システム。

24) 請求項22に記載のシステムにおいて、データ交換役割が、コピイ、移動、共用および位置づけオペレーションを含むことを特徴とするマルチ処理システム。

25) ユーザの入力のあるものが第１のプロセスに導かれ、ユーザの入力の他のものが第２のプロセスに導かれるマルチ処理システムにおいて、マッチメーカのサポートを得て第１のプロセスと第２のプロセスとの間でユーザ主導のデータ交換を行う方法が、

（A） ユーザが、第１のプロセスがデータ交換オペレーションにおいて果すべき役割を規定した入力を第１のプロセスに提供し、

（B） 第１のプロセスが、第１のプロセスに対して規定された役割の指示をマッチメーカに通知し、

（C） ユーザが、第２のプロセスがデータ交換オペレーションにおいて果すべき役割を指示する入力を第２のプロセスに提供し、

（D） 第２のプロセスが、第２のプロセスに対して規定された役割の指示をマッチメーカに通知し、

（E） 第２のプロセスに対して規定された役割が第１のプロセスに対して規定された役割と適合することをマッチメーカが検出し、

（F） マッチメーカが第１と第２のプロセスの間で通信を設定する

過程を含むことを特徴とするデータ交換の方法。

26) 請求項25に記載の方法において、前記の通信を設定する過程が各プロセスに適合したことの検出を通知し、かつ各プロセスに、該プロセスが他のプロセスとの通信において用いうるコードを提供することを特徴とするデータ交換の方法。

27) 請求項25に記載の方法においてデータ交換の役割が、コピイ、移動、共用および位置づけオペレーションを含むことを特徴とするデータ交換の方法。

3．〔発明の詳細な説明〕

産業上の利用分野

本発明はカストマイズ可能のコンピュータシステムに関し、特に種々のプログラム間でデータ資源を共用するメカニズムを備えたシステムに関する。

従来技術及び発明が解決すべき課題

データ処理システムにおいて繰返して起る多数の主要で一般的な問題領域があり、これら問題領域はデータおよび情報処理アプリケーションのタイプの範囲やユーザの数やタイプの増加に伴い現代のシステムにおいて益々注目されてきている。これらの領域は特に均一なシステムや環境におけるアプリケーション、データの統合、新しいアプリケーションとデータタイプとを既存のアプリケーションやデータタイプと統合させるために、新しいアプリケーションやデータタイプを既存システムに追加できる能力、アプリケーションを更新しうる能力、特にビジネス上の利害やデータの応

用規模が国際的になるにつれてプログラムやデータを一方の言語から他方の言語に変換できる能力、種々のユーザがデータを共用できること、特にタイムリに特定のデータを共用することにより各ユーザがデータの特定ボデイの最新バージョンを有することができること、および一方のフォーム(form)あるいはデータ構造から他方のフォームまたは構造にデータを転換即ち交換しうる能力を含む。

ユーザはグラフをドキュメントに含めたいと思うことがある。ユーザがグラフをドキュメントの文脈内から編集できるようにする既存の一方法はドキュメントのエデイタのグラフ編集機能を強化することである。この方法は、(例えば音声の注釈のように)さらに新しいタイプのデータをドキュメントに統合するために、ドキュメントエデイタを再度修正する必要があるという欠点を有する。事実、各エデイタは各々の新しいタイプのデータを扱うよう個々に拡張されねばならない。

従来技術のあるシステムは、全てのデータタイプが「オブジェクト」と称される標準化したデータ構造に常駐するオブジェクトベースのシステムを構成することによりこれらの問題を解決しようと試みて来たが、この従来技術のシステムは全体的に、これらの問題を十分に解決することに失敗した。特に、従来技術のシステムは一般的に２種類の一方に該当する。第１の、古い方の種類のシステムにおいては、データタイプや応用プログラムに対する制約が殆んど無く、その結果、新しいアプリケーションやデータタイプを追加することは容易であるものの統合したシステムとユーザ環境を提供することは困難で、かつユーザとデータタイプの間でデータ通信することが極めて難しい。第２の、より最近の種類のシステムにおいては、例えばオブジェクトベースのシステムであって、定義したシステム環境を提供しよと試みてきた。この方法では、ユーザおよびデータタイプ間での通信できる能力を向上させる統合されたシステムとユーザ環境を提供する。しかしながら、前記種類のシステムに対して繰返し起る問題はシステム

の定義自体および定義されたオブジェクトベースのシステムを管理する上で必要とされるオペレーティングシステムのタイアプログラムとが、もしそれらが初期に検討され、かつ定義されたアプリケーションとデータタイアとに適合しない場合新しいアプリケーションやデータタイプを追加する能力を阻害することである。

<u>課題を解決する手段</u>

本発明は、タイプされたオブジェクト、オブジェクトマネージャおよび適当なオブジェクトマネージャを呼び出しオブジェクトに対してオペレーションを実行する汎用化された呼び出しメカニズムとにより高度に統合されていて、しかも拡張可能のシステムを提供する。

本発明によるシステムは中央のオペレーティングシステムタイアのオブジェクト管理システムは利用しないが、オペレーティングシステムタイアのタスクおよび多くの汎用化したアプリケーションプログラムタイアのタスクを実行するあるグループのオブジェクト管理データ構造と複数の、一般的ルーチンの「パック」とを提供する。ルーチン「パック」は、オブジェクト管理データ構造に対してオペレートすることにより、オブジェクトマネージャがオブジェクト管理機能を実行するために全てのオブジェクトマネージャがアクセス可能で、かつ使用しうる一般的なオブジェクト管理ルーチンのパックを含む。この方法により、単にオブジェクトマネージャを設置し、そのオブジェクトマネージャを用いてオブジェクト管理データ構造において適当なエントリを生成することにより新しいオブジェクトタイプおよび新しいオブジェクトマネージャあるいはアプリケーションプログラムを既存のシステムに容易に追加しうる。

オブジェクト管理に関係したデータ構造の間には、個々のオブジェクトおよびオブジェクトの間のリンクを管理するために使用するオブジェクトカタログと、オブジェクトとオブジェクトに対して実行するオペレーションとを対応するオブジェクトマネージャに関係づけるために使用するオブジェクトマネージャテーブルと、オブジェクトの

作成に使用されるオブジェクトアロトタイアテーブルとがある。オブジェクトカタログは本システムに　駐する全てのオブジェクトのオブジェクトテーブルを含む。またオブジェクトカタログは、オブジェクトテーブルにおけるいずれかのオブジェクトへ、あるいはそこからの各リンクのレコードを有するリンクテーブルを含む。

オブジェクトマネージャテーブルは複数のオブジェクトマネージャが、各オブジェクトタイアに対する一次オブジェクトマネージャを含む、いずれか所定のオブジェクトタイアをオペレートするようにさせる。特定のオブジェクトに対してオペレートするよう呼び出された特定のオブジェクトマネージャは実行すべきオペレーションのタイアによって決まり、あるオブジェクトマネージャは1タイア以上のオブジェクトをオペレートしうる。オブジェクトタイア、実行すべきオペレーションおよび対応するオブジェクトマネージャとの間の関係づけはオブジェクトマネージャテーブルを介して実行される。即ち、ユーザが所定のオブジェ

(6) クトに対してオペレーションを実行することを選択すると、オブジェクト管理ルーチンは、そのオブジェクトタイプの対応するエントリとオペレーションをオブジェクトマネージャテーブルから読取り、呼び出すべき対応のオブジェクトマネージャを検出する。

オブジェクトタイアテーブルは新しいオブジェクトの作成に使用される情報を含む。オブジェクトタイアテーブルは本システムに設置された各タイアのオブジェクトの記憶されたアロトタイアコピイをアクセスする手段を備供する。いずれかの所定のタイアの新しいオブジェクトは、オブジェクトのアロトタイアコピイのコピイを作ることにより任意に作成することができ、アロトタイアオブジェクトのコピイが次いで、ユーザにより任意に修正あるいはオペレートされる新しいオブジェクトとなる。プロファイルエデイタを用いて新しいオブジェクト用の対応する新しいプロファイルを作成し、かつ必要に応じて基本的なプロファイルのコピイを修正し、オブジェクトの修正された

特性を反映させることができる。

また、本発明のシステムは、例えばオブジェクトあるいはデータ構造の部分のようなデータ構造をリンク即ち接続させる手段を備供する。リンキング(linking)は、また、一方のデータ構造から他のデータ構造へデータあるいは情報をダイナミックにコピイできるようにすることによって、宛て先のデータ構造に、原始データ構造に常駐する、リンクされたデータの最新のバージョンを提供することができる。

リンクは、「子供」のオブジェクトと称される一方のオブジェクトを「親」のオブジェクトと称する別のオブジェクトに接続する手段と見做してよい。子供オブジェクトを親オブジェクトに連結する他に、リンクはまた子供のオブジェクトのデータの一部を親のオブジェクトへリンクするために使用できる。子供のオブジェクトから親のオブジェクトへのデータのリンキングは、リンクされるデータは親オブジェクトの一体部分となるのではなくむしろ子供のオブジェクトの一部のままであると

いう点において、一方のオブジェクトから他方のオブジェクトへデータをコピイすることは区別される。リンクされたデータは子供オブジェクト内部のみおよび子供オブジェクトタイアと指定されたオブジェクトマネージャのみにより編集できる。

データは動的にも静的にもリンクできる。動的リンク連結の場合、データの現在のバージョンが子供オブジェクトから読取られ、リンクが「更新」される毎に親オブジェクトに提供される。前述のように、動的リンクは例えば、親オブジェクトがオペレートされる、即ち開放され、表示され、編集され、あるいはプリントされる毎に更新しうる。また、リンクの更新は周期的間隔で開始され、あるいは子供オブジェクトに対するオペレーションに固定されることによって、子供オブジェクトが何らかの要領で修正されたり、オペレートされると常に更新が発生する。

本発明によるシステムは、異なるタイアのデータ構造の間でデータの交換を行うための改良されたシステムを提供する。データ交換メカニズムは

第１の局面においては、所定のデータ構造に対する各オブジェクトマネージャが、データが交換されているデータ構造のタイプに応じて１個以上の種々の交換フォーマットを使用しうる、複数のデータ交換フォーマットを提供する。本発明の実施例においては、本システムは３クラスの交換フォーマットを提供する。第１のクラスは、種々のタイプのデータ構造と種々の内部フォーマットとの間のデータ交換に使用される一般的なフォーマットを含む。第２のクラスは同じタイプのデータ構造であるが内部データフォーマットの異なるデータ構造の間のデータ交換に使用されるカテゴリ特定のフォーマットを含み、第３のクラスは同じタイプで同じ内部フォーマットのデータ構造の間のデータ交換用の専用フォーマットを含む。

第２の局面において、本発明によるデータ交換メカニズムは、２個のデータ構造のオブジェクトマネージャが利用可能の交換フォーマットに関して通信でき、データ交換用フォーマットの選択を調整するマッチメーカ機構を提供する。

(7)

本発明の目的は、種々のアプリケーションプログラムを統合する開放型のフレームワークを提供することである。この点に関して、「開放型」とは既存のアプリケーションの修正を必要とすることなく新しいアプリケーションを既存のアプリケーションと統合すべきことを意味する。

本発明の別の目的は、アプリケーションが基本的に独立したままでありうるものの、依然として相互に効果的に通信、かつ協働できるシステムを提供することである。

本発明のその他の特徴、目的および利点は本発明の実施例の以下の説明を読み、かつ図面を検討した後は当該技術分野の専門家には理解される。

実施例

以下の説明は、本発明の現在の好適な実施例を組み入れたコンピュータシステムの構成とオペレーションとに関するものである。

詳細な説明の概要

(8)

19.2.11. Apoptinish()－オペレーションを停止する

19.3. マッチメーカオペレーション

19.3.1. Apmmreserve()－オペレーションに対するマッチメーカを予約する

19.3.2. Apmmpost()－マッチメーカに対するオペレーションを通知する

19.3.3. Apmmconnect()－一致したアプリケーションに接続する

19.4. リンク交換

19.4.1. LNXpinit()－リンクストリームの構成を開始する

19.4.2. LNXprestart()－ストリームをリンクにリセットする

19.4.3. LNXplink()－リンクをストリームに入れる

19.4.4. LNXpnewlink()－新しいリンクをストリームに入れる

19.4.5. LNXptinish()－リンクストリームの構成を終了する

19.4.6. LNXginit()－リンクストリームの読取りを開始する

19.4.7. LNXgrestart()－ストリームをリンクにリセットする

19.4.8. LNXglink()－次のリンクをストリームで取得(GET)する

19.4.9. LNXgpeek()－ストリームで次のリンクをルックアップする

19.4.10. LNXgskip()－ストリームでの次のリンクをスキップする

19.4.11. LNXgfinish()－リンクストリームの読取りを終了する

19.5. テキストの交換

19.5.1. TXXpinit()－テキストストリームの構成を開始する

19.5.2. TXXprestart()－テキストにストリームをリセットする

19.5.3. TXXpstring()－テキストストリングをストリームに入れる

19.5.4. TXXpchar()－単一キャラクタをストリームに入れる

19.5.5. TXXpfont()－現在のフォントを変更する

19.5.6. TXXpattrs()-テキストの属性をセットする

19.5.7. TXXpdiacritic()－区別的発音符を挿入する

19.5.8. TXXpstrikethru()－ストライクスルー(Strike－Thru)キャラクタをセットする

19.5.9. TXXpscript()－スクリプトオフセットをセットする

19.5.10. TXXpverticalf()－垂直降下移動指令を出す

19.5.11. TXXphorizontalf()－水平移動指令を出す

19.5.12. TXXpspacing()－行間間隔指令を出す

19.5.13. TXXplanguage()－言語変更指令を出す

19.5.14. TXXplink()－リンクをストリームに入れる

19.5.15. TXXpfinish()－テキストストリームの構成を終了する

19.5.16. TXXginit()－テキストストリームの読取りを開始する

19.5.17. TXXgrestart()－ストリームをバニラ(Vanilla)テキストにリセットする

19.5.18. TXXgnext()－データの次のタイプを入力ストリームで取得する

19.5.19. TXXgstringsize()－次のストリングの長さを取得する

19.5.20. TXXgstring()－ストリームから次のテキストストリングを取得する

19.5.21. TXXgchar()－ストリームから次のキャラクタを取得する

19.5.22. TXXgfont()－現在のフォントを取得する

19.5.23. TXXgattrs()－テキストの属性を取得する

19.5.24. TXXgdiacritic()－区別的発音符

報を取得する

20.3.6. RESptfinfo()－ファイルに関する情報を入れる

20.3.7. RESavail()－未使用の資源ＩＤを取得する

20.3.8. RESgtnext()－次の資源情報を取得する

20.3.9. RESgtprev()－以前の資源情報を取得する

20.3.10. RESchockpt()－資源ファイル更新のチェックポイント

20.3.11. RESrevert()－最後のチェックポイントへ復帰する

20.3.12. RESfreeze()-資源ファイルバージョンを凍結する

20.3.13. RESgtfver()-資源ファイルバージョン番号を取得する

20.3.14. RESptfver()-資源ファイルバージョン番号を入れる

20.4. 資源編集機能

20.5.3. RESixadd()－資源インデックスエントリを追加する

20.5.4. RESixdelete()－資源インデックスエントリを削除する

20.5.5. RESixfini()－資源インデックスの構成を終了する

20.5.6. RESixlookup()－資源インデックスエントリをルックアップする

20.5.7. RESixslookup()－USHORTエントリをルックアップする

20.5.8. RESixllookup()－ULONGエントリをルックアップする

20.5.9. RESixslookup()－ストリングエントリをルックアップする

20.5.10. RESixufind()－USHORTエントリを見つける

20.5.11. RESixffind()－ULONGエントリを見つける

20.5.12. RESixsfind()－ストリングエントリを見つける

(11) 20.4.1. RESrdcur()－資源の現在のバージョンを読取る

20.4.2. RESgtcur()－現在の資源情報を取得する

20.4.3. REScreate()－資源を作成する

20.4.4. RESwrite()－資源を書込む

20.4.5. RESrewrite()－資源を重ね書きする

20.4.6. RESptinfo()－資源についての情報を入れる

20.4.7. RESmove()－資源を新しい位置へ移す

20.4.8. RESptnum()－資源の番号を付け直す

20.4.9. RESmerge()－資源リストをファイルに併合する

20.4.10. RESdelete()－資源を削除する

20.5. 資源インデックス機能

20.5.1. RESixinit()－資源インデックスの構成を開始する

20.5.2. RESixprep()－既存インデックスの修正を始める

20.5.13. RESixget()－資源インデックスでエントリを取得する

20.5.14. RESixindex()－資源インデックスでエントリを取得する

20.5.15. RESixinfo()－資源インデックスに関する情報を取得する

20.6. バッチスタイルでの資源の作成

20.6.1. RESecreate()－バッチスタイルの資源ファイルを作成する

20.6.2. RESeadd()－資源ファイルへ資源を追加する

20.6.3. RESeclose()－資源ファイル構成を終了する

20.7. ユーザプロファイルサポート機能

20.7.1. RESptcusfud()…セストマイゼーションＩＤをセットする

20.7.2. RESaupedif()…プロファイルで資源を編集する。

1．概念

1.1. オブジェクトとオブジェクトマネージャ

本発明のシステムは「オブジェクトベースの」語に記述される一般的な種類のシステムの１つである。即ち、情報は「オブジェクト」と称される構造に記憶される。現在好適な実施例の実現において、ほとんどのオブジェクトはそれぞれ従来のコンピュータファイルシステムの１個以上のファイルに対応する。

さらにオブジェクトに関して、本発明によるシステムはフォルダとして前述したタイアを含む基本的には無限のオブジェクト「タイア」の使用を可能とし、本システムにより実行すべきデータ、情報あるいは演算の各形態に対してあるタイアのオブジェクトがある。即ち、本発明によるシステムはオブジェクトとシステムとの間で最小のインタフェースのみ定義し、いずれかのオブジェクトの内部構造あるいは形態は定義しない。そのため、本発明によるシステム内のオブジェクトはデータ、プログラムあるいはその他の情報用の汎用コンテナとして見做してよく、内部構造即ち特定のオブジェクトの形態は実行すべきオペレーションの要

(12) 件あ いは記憶すべきデータあるいは情報のタイアにより定義される。

オブジェクトに作用(オペレート)するプログラムは「オブジェクトマネージャ」として知られ、あるいは場合によって「エディタ」、「アプリケーションプログラム」あるいは「アプリケーション」と称される。「アプリケーション」という用語はまた、単一のオブジェクトタイアをオペレートするオブジェクトマネージャのコレクション(集まり)を参照するために使用される。各タイアのオブジェクトは、そのタイアのオブジェクトに記憶されているデータあるいは情報をオペレートする一次手段として構成あるいは意図した少なくとも１個のオブジェクトマネージャを関連させている。例えば、システムは、ワードプロセッシングに対して「ドキュメントタイア」のオブジェクトをサポートでき、そのオブジェクトタイアに関連したワードプロセッシングオブジェクトマネージャがある。同様に、「データベースタイア」のオブジェクトは、データベースタイアのオブジェクトに記憶したデ

ータに(又はデータで)オペレートする一次手段であるデータベースオブジェクトマネージャを関連させている。

しかしながら、特定タイアのオブジェクトに対してオペレートするオブジェクトマネージャはそのオブジェクトタイア用の一次オブジェクトマネージャに限定されないことに注目すべきである。さらに特定のオブジェクトに対してオペレートするよう呼び出された特定のオブジェクトマネージャは実行すべきオペレーションのタイアによって変わる。本発明によるシステムはいずれか所定のオブジェクトタイアに対してオペレートする複数のオブジェクトマネージャを提供し、あるオブジェクトマネージャ、例えばあるユーティリティは１個以上のタイアのオブジェクトに対してオペレートしうる一次オブジェクトマネージャは、もしユーザが異なったプログラムを選定していないとすれば、特定のオブジェクトタイアに対してオペレーションすべく省略値により呼び出される単なるオブジェクトマネージャである。

典型的には、オブジェクトマネージャは単一タイアのオブジェクトのデータに対してオペレートするが、ある場合にはプログラムが１つのオブジェクトタイア以上のオブジェクトマネージャとなるよう配置することが望ましい。さらに、オブジェクトデータ(例えばファイルコピイ)を解釈しないオペレーションを実行せず、したがって各種タイアのオブジェクトに対して用いうる各種のユーティリティプログラムがある。

1.1.1. オブジェクトタイア:フォルダ

フォルダは編成ツールとして用いられる。フォルダタイアのオブジェクトはそれが含んでいるリンクに対して主として用いられる。例えば、グループのオブジェクトは、全てを単一のフォルダオブジェクトに連結することにより論理的に相互に関連づけることができる。いずれのオブジェクトと同様に、フォルダ自体をフォルダに連結することができる。各フォルダには、フォルダの目的についてのユーザのコメント、フォルダのオブジェクトに対して何をすべきかの指令等の情報を含み

うるファイルが関連している。そのようなデータが無い場合も多いので、典型的なフォルダの有意性は単にそれに連結したオブジェクトのリストのみであり、この　合、基本的にはフォルダにあるのはオブジェクトカタログでのエントリ、主としてリンクテーブルのエントリのみである。

フォルダは、リンクがいずれのデータにもはめ込まれていないのでリンクマーカを何ら必要としない。フォルダのリンク間の順序はリンクマーカがはめ込まれている順序によって決められるのでなく、リンクテーブルに記憶されているリンクＩＤの値により決定される。

フォルダは、それがリンクを有しているオブジェクトからはデータを何ら連結しない(即ち、フォルダはリンク・パラレル・ファイルを何ら必要としない)。フォルダは全てのオブジェクトの間の関係を示すリンクを正に用いるということである。

本システムの各ユーザは一次フォルダを有する。ユーザがシステムの資源へのアクセスを得るのはこの一次手段によるものである。典型的に各ユーザの一次フォルダに連結され、各ユーザにある種の共通のシステム資源へのアクセスを提供する一次システムフォルダがある。さらに、ユーザの一次フォルダの全てはシステムフォルダに連結されている。

1.2. リンク

システムの情報は主としてオブジェクトの内部に含まれているが、オブジェクトの方は「リンク」と称される機構を介して相互に関連している。リンクは概念的には、「子供」オブジェクトと称せられる一方のオブジェクトが、「親」オブジェクトと称される別のオブジェクトに「接続」されるようにする手段と見做すことができる。各オブジェクトはある意味では、各オブジェクトが常に少なくとも１個のリンクを連結し、そのリンクを介して少なくとも１個の親のオブジェクトに連結するという点で子供のオブジェクトである。この点については、以下に説明するように、各ユーザは少なくとも１個の親オブジェクトを有し、該オブジェクトに対して、前記ユーザに関連したその他全てのオブジェクトが直接的あるいは間接的にリンクされている。「親」とか「子供」という用語はリンクの方向を指すのであって、リンクされたオブジェクト間での階層を意味するものではない。

以下の説明で述べるように、リンクはまた子供オブジェクトの一部を親オブジェクトに、かつ子供オブジェクト全体をリンクするために使用し得る。さらに、いずれの数のオブジェクトや、オブジェクトの部分をリンクを介して相互に連結することができ、かつリンクの方向は階層的に限定されはしない。

1.3. プロファイル

プロファイルは、例えばシステム、オブジェクトあるいはリンクのような何かについてユーザが見ることのできる情報である。オブジェクトプロファイルに含まれる情報はオブジェクトタイプによって左右される。ドキュメントタイプのオブジェクトに対して、オブジェクトプロファイルは、例えばフオントリスト、グローバルフォーマット情報およびプリンティングパラメータのようなオブジェクト自体に記憶された情報と共にオブジェクトテーブルに記憶された何らかの情報を含む。

1.4. 資源

「資源」とは、プログラムが使用するが、プログラムで実行可能のコードの一部としては記憶されないデータである。そのようなデータの例はアイコン、言語依存のテキスト、メッセージ、テキストフオント、フォーム、日付および時間並びにプレゼンテーションフォーマットを含む。したがって、資源はそのようなプログラムデータを、それが関連したプログラムから分離し、かつ独立して記憶する手段である。

プログラムデータを資源に入れることにより、その中の情報をプログラムが実行可能のコードに影響を与えることなくカストマイズ即ち変更することができるようにする。例えば、ワードプロセッシングデータベースおよびスプレッドシートプログラムの英語、仏語および独語バージョンは単に、対応する資源の英語、仏語および独語バージョンを提供することにより生成することができる。ア

ログラムの各バージョンにおいては、関連した資源に常駐するプログラムデータのみを変えればよく、プログラムの実行可能のコードは不変のままである。さらに、資源を使用することによりオブジェクトマネージャとその他のプログラムが共通のプログラムデータを共用できるようにし、そのため個々のオブジェクトマネージャとその他のプログラムのサイズと記憶要件を減少させる。

また資源にデータを記憶することにより個々のユーザに対してプログラムの外観とオペレーションを容易にカストマイゼーションできるようにする。資源のユーザ特定のカストマイズしたバージョンをユーザのユーザプロファイルに記憶することができる。ユーザプロファイルに介在するいずれかのカストマイズした資源を自動的にそれらの標準的なカウンタパート(相当物)と必要に応じて自動的に(かつ資源を用いてプログラムに見えないようにして)代替させる機構が提供される。

1.5.オペレーティングシステムとルーチンパック

コンピュータシステムは典型的には、オペレー

ションプログラマが現在および将来にわたって希望する全ての機能を設計者が想定するプログラムの一体構造として設計されている。オペレーティングシステムが相対的にモノリシック的で、かつ複雑であり、その結果オペレーティングシステムを構成するプログラムと相互関係があるため、オペレーティングシステムを修正したり、変更したり、更新したり別の特徴を追加することは相対的に困難である。

本発明によるオペレーティングシステムは、まずオペレーティングシステムにより実行される実際の機能とサービスとが最小に低減されるという点で従来のオペレーティングシステムと相違する。後述のように、本発明によるオペレーティングシステムは基本的にはプロセス、イベントおよび記憶管理機能と、物理的アドレス変換論理のみを実行する。

第2に、通　オペレーティングシステムにより実行されるその他の機能やサービスはアプリケーションプログラム自体で通常実行される多くの機

(14) ティングシステムと称されるプログラムのグループを含む。オペレーティングシステムは、システムの全体オペレーションを制御し、かつ促進し、該システムのユーザに対して選定した基本的で一般的なサービスオペレーションを提供するプログラムのコレクションと見做すことができる。オペレーティングシステムのそのようなサービスやオペレーションの例としてはプロセス管理、イベント管理、記憶管理、入出力操作、物理的アドレス変換論理、図形／テキストおよびディスプレイ(表示)操作、ファイルアクセスおよび管理オペレーションおよび数学的演算を含む。

オペレーティングシステムの従来からの見解では特定のアプリケーションプログラムに対して特定的あるいは一意的でない全ての機能やオペレーションはオペレーティングシステムにより実行されるべきである。さらに、伝統的な見方ではオペレーティングシステムを緊密に集積化し、かつプログラムの相互依存グループと見做しており、オペレーティングシステムは一般的に、アプリケー

能やオペレーションと共にルーチンのライブラリによって実行される。「パック」と本明細書で称するルーチンのライブラリはオペレーティングシステムとアプリケーションプログラムの双方から独立して存在している。以下の説明で述べるように、パック内に含まれたルーチンは全体的な広範の適用性を有し、かつオペレーティングシステムとアプリケーションプログラムのサポート機能の双方に対して有用なるよう構成した、短くて、一般的で、単一目的のルーチンから構成されている。パックルーチンにより実行されるサービスと機能とは限定的ではないが、入出力操作、図形／テキストおよびディスプレイ操作、ファイルアクセスおよび管理オペレーションおよび数学的演算を含む。

また後述のように、各パックは、それらが実行するタイプあるいはそれらがオペレートするオブジェクトのタイプとにより関係づけられる一組のルーチン即ち機能を含んでいる。さらに、所定のパックの全てのルーチンには均一なインタフェー

スあるいは呼び出しおよび復帰シーケンスを備えており、かつパックは特にパックヘッダファイルと制御ブロック即ちパックが制御するオブジェクトに関する情報のブロックに関して一定のフォーマットに合致する必要がある。

オペレーティングシステムの多様なサービスや機能およびアプリケーションプログラム内で通常提供される多くのオペレーションに対してルーチンのパックを用いることによりシステムのアーキテクチヤと構成のフレキシビリティを大きく向上させる。機能およびサービスは容易かつ迅速に、さらにシステムのその他の機能を阻害するか、さもなければ圧迫することなく追加あるいは変更ができる。さらに、パックを使用することにより、通常個々の各アプリケーションプログラムに組み入れられる多くのオペレーションや機能が多くのアプリケーションプログラムにより共用されうるという点でアプリケーションプログラムのサイズや複雑さを低減させる。また標準化したインタフェースを備えたルーチンパックを使用すること

(15) により、多くのオペレーション、およびシステム、ユーザおよび他のアプリケーションプログラムに対するインタフェースが定義され、標準化されたルーチンパックインタフェースにより定義され、かつ該インタフェースに合致するようにされる点で種々のアプリケーションの統合を促進させる。

好適な実施例においては、これらのルーチンは共用されたサブルーチンライブラリに記憶され、実行時間に必要に応じ動的(ダイナミック)にリンクされる。ある場合には、パックライブラリからのルーチンのコピイをプログラムの実行可能コードに含めることが望ましいが、この方法はアプリケーションプログラムの物理的サイズを増大させる。

以下の説明においてある種の機能を実行するルーチンを参照する場合、該ルーチンはその機能を直接独自で、あるいはシステム呼び出しを行うか、あるいは別のプロセスと一連のプロセス間通信を行うことにより達成することを理解すべきである。例えば、APPACKルーチンAPinvoke()は、最終的に

システム呼び出しを行いその結果オブジェクトマネージャを呼び出すことになるがアプリケーションマネージャプロセスと通信することによりオブジェクトマネージャを呼び出す。

2、アーキテクチヤ上の概観

本発明はタイプしたオブジェクトの処理(マニプュレーション)を含む。種々のオブジェクトが、例えば以下に述べるような情報の種々形態を示すように構成されている。ドキュメントオブジェクトはテキストとそれに関連するフォーマット化情報を表示する。スプレッドシートオブジェクトは数学的モデル化情報を表示する。イメージオブジェクトは写真タイプのピクチャを表示する。

各タイプのオブジェクトに対して、そのタイプのオブジェクトに対してオペレートしうる1個以上の「オブジェクトマネージャ」がある。オブジェクトマネージャは大略アプリケーションプログラムに対応する。例えば、スプレッドシートプログラムがある。これはオペレーティングシステムのプログラムと区別されて「アプリケーション」と理

解されうる。また、「スプレッドシート」タイプのオブジェクトの「マネージャ」あるいは「エディタ」とも理解しうる。

オブジェクトの構造と解釈とを定義するのはオブジェクトのオブジェクトマネージャであるので、オブジェクトタイプのコレクションは拡張可能である。既存タイプの情報は、それぞれ既存のタイプが相互に統合されているのと同じで、各々が従来タイプの情報と統合しうる。新しいタイプのオブジェクトを操作しうるプログラム(即ち、オブジェクトマネージャ)を追加することにより新しいオブジェクトタイプを本システムに追加することができる。新しいタイプのオブジェクトを操作しうる能力は一回実現されればよい。新しいオブジェクトマネージャを追加すると、新しいタイプのオブジェクトは、以前に存在のオブジェクトマネージャを修正することなく前に存在のオブジェクトへリンクしたり、データを交換したりでき、あるいは統合されたようになる。

拡張可能の統合を促進するということは、各オ

プジェクトタイプに対して、最　的なセットのリクエストを全てサポートするオブジェクトマネージャがあるということである。さらに、これらのリクエストはアプリケーションとは独立した単一の標準的プロトコルによりオブジェクトマネージャの間で通信される。この態様に絶対的に固執する必要はない。しかしながら例外の数が増加するにつれて、データ統合に対するこの方法から得られる利点が減少していく。

標準的なリクエストやプロトコルが十分な程度の統合を提供しない場合が発生すれば、専用リクエストやプロトコルを定義して標準的なセットと共存するようにできる。このように、この拡張可能の方法は、適当に応じ緊密な連結を阻止することなく最低レベルの統合のみを定義する。専用プロトコルが適当である場合の一例はスプレッドシートアプリケーションとグラフィング(図示)アプリケーションとの間の通信である。

全てのオブジェクトマネージャに対して利用可能の１組のアプリケーション統合サービスがある。

これらのサービスはいずれか特定のタイプのオブジェクトの「知識」を何ら具体化しているものではない。むしろ前記サービスは各種タイプのオブジェクトの操作を調整するためにオブジェクトマネージャが使用するものである。特に、これらのサービスは適当なオブジェクトタイプに特定的な「知識」を具体化するオブジェクトマネージャをサポートしやすくする。

アプリケーション統合サービスの名称の由来は、それらが個々のアプリケーション(即ち、オブジェクトマネージャ)がユーザに対して他のアプリケーションのそれとデータのオペレーションおよび操作とを統合しうるように見せかける機構であるためである。ユーザに対しては、テキストとピクチャの双方を有する時事通信の１頁をディスプレイすることは、ユーザにはドキュメントとして知られる単一のエンティティでピクチャとテキストとが統合されたことを示す。本発明によれば、この効果は、アプリケーション統合サービスを用いることにより調整された、２つの異なるオブジェ

クトマネージャのオペレーションにより達成される。時事通信は、テキストと関連のフォーマット化情報とを記憶するよう構成されたオブジェクトの１タイプであるタイプドキュメントのオブジェクトに記憶される。この例の特定のドキュメントオブジェクトは、写真タイプのピクチャを記憶するよう構成されたオブジェクトである「イメージ」タイプの個別のオブジェクトに対するリンクを含む。当該ページの表示は、テキストを検索し表示しているドキュメントオブジェクトマネージャと、ピクチャを検索し表示しているイメージオブジェクトマネージャとにより達成される。ピクチャに対するリンクと必要なオペレーション(表示)とを記載する情報は、アプリケーション統合サービスの助けをかりてドキュメントオブジェクトマネージャからイメージオブジェクトマネージャまで通信される。

本発明の実施例によるコンピュータシステムは、複数のプログラムを同時に有効に実行する機能を含んでいる。これはマルチタスキングあるいはマルチプロセッシングとして知られていて、同時に実行するプログラムの各々はタスクあるいはプロセスとして知られている。

本発明のシステムにおいては、アプリケーションマネージャがある。アプリケーションマネージャはオブジェクトマネージャの処理の基となる。即ち一般的には、オブジェクトマネージャは相互に対して同僚であって、アプリケーションマネージャの子供である。

アプリケーション統合サービスは部分的にはアプリケーションマネージャによって、また部分的には一組の共通のサブルーチン即ち機能によって提供される。オブジェクトマネージャの原始コードの本体の組み込みうる１組の機能呼び出しが利用しうる。この組の機能はAPPACKとして知られている。APPACK機能呼び出しは、それによりオブジェクトマネージャがアプリケーションマネージャのサービスを利用するメカニズムである。一般的に、APPACK機能はアプリケーションマネージャプロセスへプロセス間メッセージを送り、かつそこから

プロセス間メッセージを受取る。APPACK機能自体は共用のサブルーチンライブラリから呼び出し可能であって、あるいはプログラムリンク時にオブジェクトマネージャの実行可能本体へ直接組み入れることができる。

APPACKにより提供されるサービスは以下のカテゴリーに分類できる。即ち、呼び出し、データ交換、突き合わせ(マッチメーキング)、オブジェクト管理およびリンク管理である。

アプリケーション統合サービスの中で最も強力なサービスは「呼び出し」サービスである。これらサービスを取得するために介する最も一般的なメカニズムはAPinvoke()として知られるAPPACK機能であって、これは以下詳述する。特定の目的(例えばオブジェクトからデータをプリントするAPreprint()や、リンクにわたってデータを更新するAPrequpdate()で呼び出すためにその他のAPPACK機能も利用できる。

呼び出しサービスはいずれのオブジェクトマネージャも(例えばリンク仕様により)あるデータを

らのサービスはデータの形態と転送の手段の双方をマスクすることによって、それらを使用するアログラムはデータの意味のある中味にのみ関係すればよい。

マッチメーキング(組合わせ決定)サービスはユーザ主導のデータ転送を設定するために使用される(これはデータ交換サービスを用いて完了するオペレーションである)。これらのサービスはデータ転送元(ソース)と受信者との間の交渉を促進する。

オブジェクト管理サービスとリンク管理サービスとを使用することは、本システムのオブジェクトおよびリンク機能を利用するオブジェクトマネージャにとって必要である。オブジェクト管理サービスはオブジェクトのアクセス、ネーミングおよび操作に関する基準を定義する。リンク管理サービスはオブジェクトマネージャが、それについてオブジェクトマネージャが何ら直接の「知識」を有さないタイプのオブジェクトへのリンクを保てるようにする。

(17) 規定し、そのデータに関してオペレーションを実行できるようにし、適当なオブジェクトマネージャを識別し、かつ呼び出し、その結果呼び出されたオブジェクトマネージャが特定のタイプのデータを何ら扱うことなく希望するオペレーションが実行されるようにする。例えば、ドキュメントオブジェクトマネージャは、それぞれ同等の容易さで、かつイメージあるいはオーディオ(音声)データのいずれかを操作するよう構成することなくピクチャを表示したり、音声メッセージを再生できるようにする。

標準的なプロトコルを実行する役割を果す以外に、呼び出しサービスはまた、2個のオブジェクトマネージャの間で通信を行い、続いて各オブジェクトマネージャが専用プロトコルに基いて通信するようにするためにも利用できる。

データ交換サービスは、データに対して類似の内部表示を共用しえない2個のオブジェクトマネージャの間でデータを転送するために共通のデータタイプに対して標準的な表示を提供する。これ

3．オブジェクトとファイル

本発明のシステムにおいては、ほとんどのオブジェクトタイプに対して、各オブジェクトは(出来れば1個以上のファイルの)個別のファイルに記憶される。しかしながらこれは必要な要件でない。ファイルとしてのオブジェクトを実行することの利点は、あるオブジェクトのオペレーションはアプリケーション統合サービス(例えばオブジェクトの作成、オブジェクトのコピイ)により直接容易に実行できるということである。

本発明のシステムにおいては、インヘリタンス(inheritance)の形態は限定されている。ファイルベースのオブジェクトに対して実行しうるある種のオブジェクト管理オペレーションはAPPACKにより直接サポートされる。即ち、オブジェクトの作成、オブジェクトのコピイ、オブジェクトの名前の付け直し、オブジェクトの削除およびオブジェクトの凍結である。前記オペレーション中の1つがファイルベースのオブジェクトに対して要求されると、オブジェクトマネージャテーブルが

そのオブジェクトに対するオブジェクトマネージャがオペレーションを実行している旨指示し、その場合オブジェクトマネージャが呼び出され、オペレーションリクエストが該オブジェクトマネージャへ通される。APPACKはファイルベースのオブジェクトに対してこれらのオペレーションを実行しうるのみであるので、これらのオペレーションは非ファイルベースのオブジェクトにより引き継がれえない。

各オブジェクトタイプが別のタイプをその親として識別しうるさらに一般的なインヘリタンスを実行しうる。そのような状況においては、子供のタイプに対して特に定義されていないオペレーションは親のタイプのオブジェクトマネージャにより子供タイプのオブジェクトに対して実行しうる。これは新しいオブジェクトタイプを作成しやすくするという利点を有する。他方、インヘリタンスが生成されていないシステムはそれ自体より単純で高性能のシステムとなる。

4．データ統合

(18) データ統合を改良する本発明によるシステムの利点を理解する上で広範囲の緊密に関連し合った概念の間の区別を理解することが有用である。例えば、ユーザは(「原始(ソース)」オブジェクトの全て、あるいは一部である)既存のデータが新しい場所(以前に存在していたかあるいはデータを受取るためにのみ作成しうる「行き先(宛て先)」のオブジェクトにおける場所)へ「コピイ」「移動」あるいは「共用」オペレーションにより現われるようにでき、これらのオペレーション各々は他の2つのオペレーションとは相違している。さらに、コピイあるいは移動オペレーションは、コピイあるいは移動されたデータがその行き先(行き先オブジェクト相当のオブジェクトマネージャによりユーザからかくされている行き先)で「内部化(internalized)」あるいは「密閉化(encapsulated)」されるように実行することができる。

行き先オブジェクトへのデータの「共用」とは、データは行き先オブジェクトにあるように見えることを意味し、これは共用されたデータに対して

リンクにより達成され、共用されたデータは実際には行き先内には記憶されない。さらに、リンクは他のリンクがすでに参照しているものと同じデータを参照する。即ち、同じ記憶されたデータが「共用」されるのである。共用の効果は共用されたデータへの各リンクが更新されると(更新のタイミングは他の個所で論じられているリンク更新フラッグによって左右される)、その変更を明示することである。

「コピイする」ことはコピイされたデータの記憶と、原始データの記憶とが区別され、そのため一方が他方に影響を与えることなく変更可能であるという点で共用とは相違する。換言すれば、実際のデータの新しいコピイが作られる。さらに、コピイされたデータは2種類の要領で処理できる。即ち、行き先オブジェクトにおいて「内部化」するか、行き先オブジェクトにより「密閉化」できる。もしコピイされたデータが行き先オブジェクトで記憶しうるタイプのものであるとすれば、コピイされたデータは行き先オブジェクトにおいて内部

化され、これはデータが行き先オブジェクトのデータ構造において直接記憶されることを意味し、内部化されたデータはオブジェクトに記憶されたいずれかの他のデータとは区別できなくなる。コピイされたデータが行き先オブジェクトで記憶できないタイプのものである場合、データを記憶しうるタイプの新しいオブジェクトが作成され、この新しいオブジェクトへのリンクが行き先オブジェクトに追加される。この場合(密閉化)、行き先オブジェクトは間接的に、コピイされたデータを取容する。

各アプリケーションはデータを密閉化するリクエストに応答する必要がある。応答の最も単純な方法は、データを含むオブジェクト全体のコピイを作り、その新しく作成されたオブジェクトを参照するリンク仕様を提供するものである。この単純な方法は、密閉化すべきデータがオブジェクト全体の中の極めて小さい部分である場合は非能率的である。このように、各アプリケーションは出来るだけ少ないデータをコピイすることによりデ

ータ密閉化のリクエストに応答するように構成すべきである。換言すれば、出来ればアプリケーションは全体オブジェクトより小さいもの、好ましくは希望するデータのみをコピイすべきである。チャート(図表)アプリケーションは完全なチャートを作りうるのみであるため、チャートを密閉化するには、たとえチャートの一部のみ所望であってもチャートオブジェクト全体のコピイを必要とする。別の極端な場合、ドキュメントオブジェクトマネージャは必要とされるものの近辺までを抽出しうる。イラストレーションオブジェクトマネージャはさらに別の場合である。それは希望する領域の外に来るイラストレーションエレメントはコピイしない。それは、たとえ部分的であっても希望する領域に入るエレメントを、該エレメントを希望領域のサイズまで切り取ることなくコピイする。このように、希望する領域より外側のイラストレーションの部分は密閉化されたデータ内で表示される。

「移動」は原始データが削除されている以外コピ

(19) イすることと似たようなものである。移動の宛て先が著しいデータ損失を伴ってデータを受取りうるとすれば(例えば、宛て先がリンクをサポートしないとすれば)、ユーザはそのような警告を受け、移動オペレーションを止める機会が与えられる。

4.1. ユーザに対する宛て先オブジェクトの見掛け

前述の説明は宛て先オブジェクトが内部的に、共用されたデータ、コピイされたデータあるいは移動したデータを表示する態様に焦点を当ててきた。若干離れた論議はそのデータがユーザに提供される態様である。一旦データが内部化されると、宛て先オブジェクトに記憶された他のデータとは相違しない。ユーザに対するその見掛けも必然的に相違せず、該オブジェクトの他のデータと共に編集可能である。宛て先オブジェクトにより扱いうるタイプであるが(共用されているため)リンクされているタイプのデータは宛て先オブジェクトに実際に記憶されたデータとは区別のつかない態様でユーザに提供されうる。しかしながら、前記

データはオブジェクトに直接記憶されるのと正確に同じ要領で編集できないので、何らかの方法で事実上区別がつかないようにすることが好ましい。(それがリンクされているオブジェクトに対して外部のタイプであるか否かには関係なく)リンクされているデータは個別にリンクされたデータ上で編集オペレーションを呼び出すことにより編集される。リンクされているデータのあるものは、固有のデータと相互作用しえないほど異なったタイプであっても依然として固有のデータと共に可視的に現われる可能性がある(外部データを表示するため、宛て先のオブジェクトマネージャはAPPACKを介して、そのようなデータを表示できるオブジェクトマネージャを呼び出す)。これは「隔離された」データとして知られている。たとえ隔離されたデータがオブジェクトにおける他のデータと同じタイプのものであったとしてもデータ隔離が用いられるような状況がある。例えば、ドキュメントのピクチャと関連した見出しが単に固定位置に修正されず、かつドキュメントの他のテキス

トとフォーマット化されずに現われることがある。このように、見出しはテキストとしてでなくむしろイメージとして効果的に扱いうる。

隔離が可能でない場合、統合されたプレゼンテーション(表示)の最小の形態は「アイコンによる表示」であって、リンクされたデータの介在を示すアイコンが宛て先オブジェクトのディスプレイの適当な位置に表示される。他のオブジェクトへのリンクをサポートする全てのアプリケーションは少なくともアイコンによりディスプレイをサポートすべきである。リンクがデータ交換を指示しない場合(即ち、リンクがリンク仕様を含まない場合)アイコンによるディスプレイはリンキングアプリケーションが成しうる最良のものである。このように、フォルダオブジェクトはアイコン以外でユーザに現われることは全くない。

4.2. あるデータ統合概念を示す例

以下の例は前述したデータ統合概念のあるものを示す。

第1のドキュメントがチャートを含む。これは

チャートを記憶するに適したオブジェクトへのリンクを含むドキュメントオブジェクトにより達成される．ユーザは第２のドキュメントを作成する．ユーザは(例えばマウスを動かし、関連の如を入れることにより)第１のドキュメントでチャートを選択し、次いで「共用」キーを押す(あるいは他の手段では共用命令を呼び出す)．次いでユーザはチャートが現われるべき第２のドキュメントの位置を指し、次いで「アレース(位置指示)」キーを押す(あるいは他の手段ではアレース命令を呼び出す)．次いで、チャートは第２のドキュメントに現われる．次いでユーザは、第２のドキュメントの内部からチャートを選択し、編集キーを押し、チャートオブジェクトマネージャを呼び出す．ユーザはチャート内で変更する．次いで、ユーザは第１のドキュメントを再び開き、第２のドキュメントの内部からユーザが行ったばかりの変更を含むチャートを観察する．この例においては、チャート自体のコピイが一枚ある．共用オペレーションによりチャートへのリンクのコピイの第２のド

(20) キュメントを作成したので単一のチャートに２個のリンクがあることになる．

引続きこの例を検討する．ユーザが第２のドキュメントにおいて第１のドキュメントのそれと似たテキストのあるパラグラフを利用したいと思う．ユーザはこのテキストが現われるべき第２のドキュメントにおける個所を指しプレースキーを押す．(宛て先オペレーション(アレース)と原始オペレーション(コピイ、移動あるいは共用すること)はいずれの順番でも呼び出し可能である．次いで、ユーザは第１のドキュメントまで進み、テキストのパラグラフを選択し、コピイのキーを押す．次いでユーザは第２のドキュメントに戻り、パラグラフが希望位置に現われるか観察する．ユーザはそのパラグラフを編集する．後でユーザは第１のドキュメントに戻り、原始テキストが変ってないか、即ち第２のドキュメントでコピイしたテキストを編集することにより第１のドキュメントの原始テキストに何らかの変化をもたらしていないか観察する．

前記の例を更に継続して検討する．ユーザが第１のドキュメントにあるピクチャを第２のドキュメントに組み入れたいと思う．ユーザがこれを実行したいのは、前述のテキストのように、しかし前述のチャートのようではなく、ピクチャの見掛けの各々は他のドキュメントのピクチャを変えることなく変えうるからである．ユーザは、第１のドキュメントのピクチャを選択し、コピイのキーを押し、第２のドキュメントにおける希望位置を指し、プレースキーを押すことによりこの作業を達成する．このためピクチャを含むピクチャオブジェクトのコピイを作り、かつこの新しく作成されたピクチャオブジェクトへのリンクを第２のドキュメントにおいて作成する．

この例を１ステップ先まで引続き検討する．ユーザが第１のドキュメントのテキストの文が同一形態で常に第２のドキュメントに現われ、その文で行った何らかの変更が双方のドキュメントに現われるようにしたいと思う．ユーザはこのことを、希望する文を選択し、共用キーを押し、第２のド

キュメントにおける位置を指し、プレースキーを押すことにより達成する．第１のドキュメントを記憶するオブジェクトを識別し、かつ(ドキュメントオブジェクトマネージャが理解しうる要領で)選択されたテキストの文を識別するリンク仕様を含むリンクが第２のドキュメントで作成される．

データの「内部化」と「密閉化」との区別は前述の例を参照して表示しうる．ドキュメントオブジェクトはテキストを記憶するよう構成されたものであるため、第２のドキュメントにコピイされたテキストのパラグラフは、第２のドキュメントを記憶したドキュメントオブジェクトにおいて内部化される．このことはコピイされたテキストが前記第２のドキュメントに記憶された他のテキストと共に直接記憶されたことを意味する．対照的に、第２のドキュメントにコピイされたピクチャは密閉化される．この理由は宛て先オブジェクト(ドキュメントマネージャ)が直接ピクチャデータを記憶できないからである．チャートと文とが(コピイされるのでなくむしろ)それぞれ共用される

ので、それらはそれぞれ(内部化されるのではなくむしろ)第2のドキュメントにおいて密閉化される。

オブジェクトがデータを内部化するにつれて、このデータはオブジェクトにすでに存在しているデータと区別がつかなくなる。内部化したデータは編集でき、あるいは前に存在していたデータと共に直接操作できる。密閉化したデータを編集するために、ユーザは密閉化したデータを選択し、編集命令を出すことにより、密閉化したデータが記憶されているタイプのオブジェクトを扱うことの出来るオブジェクトマネージャを呼び出す必要がある。何故ならこの差により密閉化されたデータは典型的にはユーザに対して可視マーキングを(例えばボックスで囲んだり、文字上の属性で表示したりして)付されるからである。

前述の例においては、ユーザは必ずしもピクチャデータがテキストデータとは別に記憶されていることに気をつけておく必要がないが、ユーザは、ピクチャを編集するには新しいウインドウを開く

(21) ことになるエキストラオペレーションが必要なことを知る必要がある。

前述の例で示される別の点は、オブジェクトの全て、あるいは一部に対して前記の統合オペレーションを実行しうることである。前述のように、オブジェクトが「コピイ」され、データが「密閉化」されると、新しいオブジェクトが作成されてそのデータを記憶し、かつ新しいオブジェクトに対するリンクが宛て先オブジェクトで作成される。コピイすべくユーザが選択したデータがオブジェクトの唯一部である場合、新しく作成されたオブジェクトは(1)原始オブジェクト全体のコピイ、(2)希望する部分のみ、あるいは選択されたものより大きいが全体よりは小さい部分のコピイを含むことができ、特定の場合のデータの量はオブジェクトのタイプ、および可能性としては選択された部分と選択されない部分との間の関係によって変わる。新しいオブジェクトが選択したデータ以上のものを含む場合、リンク仕様が選択した部分を識別する。

5．リンク

5.1．リンクの更新

5.1.1．いつ更新が起るか

リンクにより子供のオブジェクトからのデータが親のオブジェクトに現われるようにする。概念的には、子供のオブジェクトにおけるリンクされたデータが修正されると、その修正は親のオブジェクトにおいて明示すべきである。直接的にリンク結合を達成する実行によりシステムに対して強度の性能要求を課しうる。

本発明によるシステムはリンクパラレルファイルと、広範囲の代替的なリンク更新状態とを用いてシステムに対するリンク結合による負荷を低減し、一方ユーザに対してリンク結合の利点を提供している。リンクパラレルファイルは親のオブジェクトが容易にアクセスしうるリンク結合のデータのコピイを記憶するための便利な倉庫である。もしこのコピイを連続的に使う場合、親オブジェクトは常に子供オブジェクトにおいてデータを正確に反映することになる。しかしながら、実際的な問題として、もしリンクパラレルファイルがある重要な時点、例えば子供のオブジェクトにおけるデータが変るときとか、あるいは親のオブジェクトに対してオペレーションが実行されるときに更新されるとすれば同じ結果を達成することができる。各リンクは、該リンクの状態に適合するよう選択しうる関連の更新状態を有する。更新状態とは、本明細書の詳細な説明において述べるようにリンクパラレルファイルにおけるエントリでのフィールドである。更新状態に対して現在定義されている値は、マニュアル(Manual)、ファーストタイム(First Time)およびダイナミック(Dynamic)である。一般的に、ユーザが親オブジェクトを観察している間に子供オブジェクトにおいては何ら変化は発生しない。このように、ほとんどのリンクはユーザが親のオブジェクトに作用する毎に、即ち「ファーストタイム」の更新状態のときにのみ更新されればよい。もしユーザがスプレッドシートで作業すると同時にスプレッドシートからリンクされた数でドキュメントを観察し

たい場合、およびユーザがドキュメントが常に現在のスプレッドシートの値を反映しているか調べたい場合、そのリンクの更新状態を「ダイナミック」にセットすべきで、このためスプレッドシートが　正される毎に更新オペレーションを発生させる。

5.1.2. リンク更新オペレーションは再帰を要す

リンク更新オペレーションは再帰を要する可能性がある。「リンク更新」に含まれたデータには「別のリンク」がはめ込まれている。この「別のリンク」における更新状態のフラッグの状態に応じて、元の更新オペレーションを完了させるためにこの「別のリンク」に対して更新オペレーションを実行する必要がある。「別のリンク」により基準化されるデータ自体は、はめ込まれたリンクを含む。このように、リンク更新オペレーションが完了することはリンクにより指示される深さまでの入れ子関係の更新オペレーションの完了を含む。

リンクとオブジェクトとが階層関係に編成されるとは限らないので、一連のシリーズが依存ルー

(22) プを形成することができる。単純な例としては、第１のオブジェクトが、第２のオブジェクトの一部を基準とした第１のリンクを含み、第２のオブジェクトが第１のリンクを含む第１のオブジェクトの部分を基準とする場合である。本発明のシステムは更新オペレーションが循環的であって、エラーを発することを検出する。代替的システムでは単に１回の循環、データが安定するまで(このようなことは発生することがない)の循環、あるいは間欠的な循環を含む。

5.2. はめ込みリンクを有するデータをコピイ

他のオブジェクトへのリンクを含むオブジェクトをコピイする場合、この参照されたデータをいかに処理するかの問題が発生する。リンクあるいはデータ自体がコピイしうる。ユーザは、例えば以下の例が示すように種々の状態において種々の結果を所望することがある。

本発明のシステムにおいては、各リンクと関連した「コピイフラッグ」がある。コピイフラッグの値が、リンクを含むオブジェクトがコピイされる

場合にリンクを処理する態様を決める。リンクが作成される場合、そのコピイフラッグの初期値は、ユーザあるいはアプリケーションが規定したリンクを作成しないならば省略値のルールによりセットされる。その結果はコピイオペレーションがユーザの期待する結果を度々達成する。コピイフラッグをセットする省略値ルールは以下の通りである。リンクされたオブジェクトがリンクを含むオブジェクト内から作成されると、コピイフラッグはリンクされたデータがコピイされるようにセットされる。さもなければ、コピイフラッグは、リンク(リンクされたデータでなく)がコピイされようにセットされる。

5.3. リンクマーカ

リンクマーカはオブジェクトのデータの本体に含まれリンクされたデータの介在を指示する。オブジェクトのデータはそのオブジェクトタイアに典型的に独特のフォーマット(オブジェクトのオブジェクトマネージャのみに「知られる」必要のあるフォーマット)に記憶されるので、リンクマス

タのフォーマットもアプリケーションに依存している。リンクマスタは少なくともオブジェクトを読取るアプリケーションがリンクＩＤの値を決定できるようにする必要がある。リンクＩＤはリンクに関する別の情報がリンクテーブルから検索できるようにする。

アプリケーション設計者は子供のオブジェクトのオブジェクトタイプをリンクマーカに含めるよう選択すればよい。このため、子供のオブジェクトのオブジェクトマネージャを呼び出す前に(子供のオブジェクトのオブジェクトマネージャを識別するに必要であるが、(タイプに関する情報を検索するアプリケーション統合サービスにより)オブジェクトテーブルの探索を排除することにより性能を向上させうる。

リンクマスタフォーマットは、リンクマスタを含むオブジェクトにおけるデータが修正されると、残りのデータに関するリンクマスタの位置は出来るだけ緊密にユーザの期待に対応するよう設計すべきである。以下の２種類の実行を検討する。即

ち(1)データファイルのどこにリンクが論理的に位置していたかを示すバイトカウントを備えたリンクＩＤのテーブル(2)前記データの関連位置にはめ込まれたリンクＩＤが続くエスケープシーケンスである。第１の例によりマーカで識別したリンクは、リンクマスタに先行するデータが追加されたり、あるいは削除されると移動しているように見える。少なくともドキュメントに関しては、第２の代案がより緊密にユーザの期待に対応する。

前節に述べる第１の実行により示すように、リンクマーカはリンクされたデータが現われるべき親のオブジェクトのデータの位置に物理的に記憶される必要はない。

5.4. リンク仕様

そこを横切ってデータが通りうる各リンクに対して、リンク仕様がある。フォルダオブジェクトへのリンクは決してリンク仕様を含むことはない。フォルダはオブジェクトの構成を作成するために使用するオブジェクトであり、かつオブジェクト自体は何ら実質的なデータを含まないため、リン

(23) ク仕様は必要でない。

リンク仕様は、リンクされたデータがリンクされたオブジェクトの全体であることを示し、あるいは特定の部分を識別することができる。リンク仕様の解釈はリンクされたオブジェクトのオブジェクトマネージャにより実行される。即ち、リンク仕様は特定のアプリケーションに係るものである。リンクされたオブジェクトが修正された後リンク仕様の解釈が出来るだけユーザの期待に対応するようリンク仕様の実現を考えることが望ましい。リンク仕様を実現する以下の方法がこの点を示す。　データベースオブジェクト用のリンク仕様は質問仕様である。このタイプのリンク仕様はユーザの期待に対応するという希望する結果を特に十分達成する。

図示のためのリンク仕様は以下の形態である。３頁１インチ下２インチ横から始まって、２インチ長さで３インチ幅の長方形の部分を用いる。これにより図示のある頁が入るウインドウを規定する。その頁のピクチャがシフトされる結果、リン

クで別のデータが規定される。

スプレッドシート用のリンク仕様は、絶対的な座標あるいは名前付き範囲のいずれかで規定されるある範囲のセルである。スプレッドシートの変更により、スプレッドシートのユーザより容易に理解される要領によりリンク仕様によりいずれのセルが規定されるか作用する(例えば、絶対的な座標が使用されると、規定された組のセルは、その範囲の左側のコラムが削除されると変化する)。

ドキュメントオブジェクト用のリンク仕様は顕著に異なる方法をとる。ドキュメントの一部がリンクすべく選定されると、ドキュメントの選定された部分は現在の位置から除去され、ドキュメント内のシェルフ(「棚」)に位置される。その部分が以下常駐していた位置には新しく作成されたシェルフへの索引が位置される。リンク仕様がこのシェルフを識別する。ドキュメントエディタがリンクされたマテリアルのようにシェルフ上のマテリアルを取り扱う。シェルフ上のマテリアルは適当な位置でドキュメントで表示されるが、シェル

フ上のマテリアルはそのシェルフでの編集オペレーションを特別に呼び出さずには編集することはできない。このように、ユーザはドキュメントのテキストのあるものを共用している別のオブジェクトにおいてドキュメントに対する変更が明示される態様を容易に理解する。シェルフの外側での変更は何ら効果なく、シェルフ上の変更はリンクを横切って見られる直接変更したデータである。

5.5. オブジェクトとリンクの凍結

オブジェクトは凍結できる。凍結されたオブジェクトは決して修正することができない。本発明によるシステムに対する組則はオブジェクトが一旦凍結されたものとしてマーキングされると、再び凍結したとマーキングされることは無いことである。コピイは凍結したオブジェクトからつくることができ、コピイは修正することができる。オブジェクトは、オブジェクトカタログのそのオブジェクトのレコードにおいて「凍結」フラッグを設計することにより凍結したものと識別される。

リンクは凍結することができる。これはリンク

の更新状態をマニュアルにセットすることにより実行される。マニュアルの更新状態のリンクにより提供されたデータは、次のときにリンクを新しいデータで更新すべきとのリクエストがなされるまで一定のままである。リンクの更新状態がマニュアルであると、リンクに関連したデータが宛て先オブジェクトのリンクパラレルファイルに記憶されるコピイとなる。このデータはユーザが更新オペレーションを要求するまでは不変のままであるが、前記の要求時データの新しいコピイが作られる。リンクの更新状態がマニュアルにセットされると、更新オペレーションが実行され、リンクパラレルファイルにデータがあることを保証する。

オブジェクトに対する連結したリンクはそのオブジェクトの修正可能性には影響しない。該リンクはその特定のリンクを通して視たオブジェクトの外観のみを連結する。前記外観を含むデータは(宛て先オブジェクト即ちリンクを含むオブジェクトである)リンクパラレルファイルに記憶され、リンクが作られているオブジェクト(原始オブジェ

(24) クト)は依然として修正可能である。

6．物理的構成(第１Ａ図、第１Ｂ図および第１Ｃ図)

第１Ａ図、第１Ｂ図および第１Ｃ図は本発明を組み込みうる情報処理システムのブロック線図である。

第１Ａ図を参照すれば、顕著なディスク記憶容量172、プリンク174およびその他の共用の周辺装置176を備えたホストコンピュータ170は複数のインテリジェントワークステーション178に接続されている。各ワークステーション178は仮想メモリをサポートするメモリ管理を備えたコンピュータを含む。ワークステーションは局部的に接続された周辺装置を含んでよいが、典型的にはホストコンピュータ170に接続された周辺装置174および176を使用する。ワークステーション178はローカルディスク記憶装置180を含んでよく、その場合仮想メモリシステムはホストコンピュータのディスク172へでなく、むしろ局部的にページング出来る。

特定の構成においては、高速(４メガビット／秒)の直列データリンク182を介して各ワークステーション178に接続されたウォングＶＳミニコンピュータで、ワークステーションは各々モトローラ(Motorola)68020プロセッサである。この構成において、ホストコンピュータはＶＳオペレーティングシステムを実行し、オブジェクトカタログとシのデータベーステーブルのあるものを管理するために汎用の関連データベースシステムを使用し、かつシステムのデータベースの残りに対してより単純なツールを用いる。(以下説明する)カタログサーバプロセッサ190がＶＳで実行する。各ディスク172は１個以上のボリュームとして構成され、その各々はオブジェクトおよびその他のファイルを記憶するために使用され、かつ対応するオブジェクトカタログを含む。一方のボリュームがシステムデータベースを含む。各ワークステーション178は必要に応じてマルチタスキングカーネル192、アプリケーションマネージャプロセス194、および１個以上のオブジェクトマネージャ196を実行

する。APPACKルーチン218とRESPACKルーチン222とかワークステーションに介在し、そこで実行している各種のオブジェクトマネージャ196により呼び出される。

代替形態(第１Ｃ図)においては、ワークステーションはローカルエリヤネットワーク154上の対等なものとして接続され、前記ネットワークには共用資源用サーバ(例えばファイルサーバ158、プリンドサーバ)および例えばミニコンピュータあるいはメインフレームコンピュータのような共用計算資源も接続されている。

さらに別の代替構成(第１Ｂ図)においては、(キーボード118とディスプレイ120とを含む)ダム(dumb)ワークステーションが(メモリスペース110と中央処理装置116とを含む)ホストコンピュータに取り付けられている。この構成において、オブジェクトマネージャとアプリケーションマネージャとは、ワークステーションでなくむしろホストコンピュータで実行される。

第１Ｂ図を参照すれば、全体的な情報処理シス

テム、即ち、プログラムにより制御され、かつシステム100のユーザにより指示されてデータあるいは情報に対してオペレーションを実行するコンピュータシステムのブロック線図が示されている。図示のように、システム100はメモリスペース110、即ちオペレーションを実行する上でシステム100までアクセス可能で且つ該システムにより使用可能のメモリスペース、を含み、該スペースにおいてユーザによる指示と制御とによって対応するデータ構造に対して各種のオペレーションを実行するためのプログラムやオペレートすべきデータ構造が常駐する。図示のように、メモリスペース110は、現在使用中のデータ構造やプログラムを記憶するためのメインメモリと、現在使用中でないデータ構造やプログラムを記憶するための、1個以上のディスクドライブのような大記憶メモリ114から構成されている。前記システム100にはまた、システムのオペレーションを実行するためにプログラムに対して応答する中央処理装置(ＣＰＵ)116が含まれている。ユーザの入力に対してキー

(25) ボード118が設けられ、かつユーザにオペレーションを可視表示するためにディスプレイが設けられている。また、システム100はシステム100へ、かつそこからの情報入出力を供給するためにテレコミュニケーション、ネットワークインタフェースおよびプリンタのような1個以上の周辺装置122を含んでもよい。これら周辺装置122のあるものはシステム100のメモリスペース110内に含めてよいことに注目すべきである。

システム100のメモリスペース110に常駐するプログラムとデータ構造とを参照すれば、これらのエレメントはユーザの指示でデータ構造に対してオペレーションを実行するプログラム124と、オペレートされるデータ構造126とを含む。また、本システムのオペレーションを導き、監視し且つ調整する機能を提供するオペレーティングシステムのプログラム128と、本システムとキーボード118、ディスプレイ120および周辺装置122との間での情報の転送を制御する入力／出力プログラム130とが含まれている。

第１Ｃ図を参照すれば、本発明を組み込みうる第２の情報処理システム(システム150)が示されている。該システム150は、単一のＣＰＵ116と、多数のユーザが共用しうるメインメモリ112との代りに、システム150が、それぞれが単一のユーザに供しうる複数のＣＰＵ116とメインメモリ112とを備える点においてシステム100と相違する。しかしながらシステム150はシステム100と同様に、現在使用されていないプログラム124を記憶する単一の大容量メモリ114とデータ構造126とを備え、単一の大容量メモリ114も本システムの全てのユーザにより共用される。

図示のように、システム150は一般的に第１Ｂ図に示すシステム100と同じ基本的エレメントを含む。しかしながら、システム150は複数の処理装置152から構成され、該処理装置の方はファイルサーバ156にバス154を介して接続され、該サーバの方はシステム150の中央に配置の大容量メモリ114に接続されている。各処理装置152、即ちワークステーションの方はＣＰＵ116と、メインメ

モリ112と入力／出力装置158とからなり、該入力／出力装置158は例えばキーボード118、ディスプレイ120および周辺装置122とを含む。

システム100におけるように、現在使用中のプログラム126とデータ構造126、オペレーティングシステムのプログラム128と入力／出力プログラム130とは各処理装置152のメインメモリ112に常駐し、入力／出力装置を介して処理装置152と対話しているユーザの指示によってオペレーションを実行する。イナクティブ(inactive)プログラム124とデータ構造126とは単一の中央に配置の大容量メモリ114に常駐し、大容量メモリ114と、処理装置152の個々のメインメモリ112との間をファイルサーバ156とバス154とにより転送される。この点に関して、ファイルサーバ156は処理装置152からのリクエストを読取ったり書込むよう応答し、データ構造126とプログラム124とを、ユーザのオペレーションにより要求されるに応じて大容量メモリ114と処理装置152との間で読取ったり書込んだりする。さらに、オペレーティングシステムプ

ログラム128と入力／出力プログラム130とは大容量メモリ114において記憶されかつ必要に応じて例えば処理装置152の開始時に大容量メモリ114から処理装置152のメインメモリ112へロードしうる。これらの目的に対して、ファイルサーバ156は例えばインテリジェントドライブシステムのような専用ファイルサーバより構成するか、あるいはシステムユーザの指示とリクエストとにより更に別のオペレーションを実行しうる汎用コンピュータから構成すればよい。

7．表示システムのあるエレメントについて

本発明のシステムのエレメントのあるものには、システムデータ構造、アプリケーションパック(APPACK)218、関連した資源パック(RESPACK)222を備えた資源がある。

システムデータ構造は本システムの管理に関係した複数のデータ構造体、その中のオブジェクトおよびオブジェクトマネージャと資源とである。システムデータ構造はシステムの一次フォルダ、システムのデータベース250および1個以上のオ

(26) ブジェクトカタログ252とを含む。第2図に示すように、システムベース250はオブジェクトタイプテーブル254、オブジェクトマネージャテーブル256、およびオブジェクトプロトタイプテーブル258とを含む。第5図に示すように、オブジェクトカタログ252はオブジェクトテーブル260、リンクテーブル262およびファイルリストテーブル264とを含む。リンク平行ファイル266も、それらが公に入手しうる情報(例えば、更新状態、表示状態および表示モードフィールド)を含む範囲まではシステムデータ構造である。

APPACK218の方は、オブジェクトマネージャの統合、オブジェクト管理、オブジェクトマネージャの呼び出しおよびオブジェクトマネージャの「突き合わせ」およびオブジェクト間でのデータの交換に対するサービスと機能とのパックからなる。

ある場合には、特定のシステムにおけるオブジェクトとファイルとは「ボリューム」に編成され、特定のオブジェクトカタログは所定のボリューム内でのオブジェクトへのエントリを含み、1個以上

のボリュームがある場合、該システムにおけるオブジェクトカタログは1個以上である。その他のシステムにおいては、オブジェクトとファイルとは単一のオブジェクトカタログに収録しうる。オブジェクトカタログはボリュームにある全てのオブジェクトの中オブジェクト識別子により割り出されたオブジェクトテーブルを含む。さらに、オブジェクトカタログはオブジェクトテーブルにエントリを有する各オブジェクトへ、あるいはそこからの各リンクに関する基本的情報を含む。

データリンクを有するオブジェクトカタログにおいて各組のオブジェクトに関してリンクパラレルファイル266がある。特定のオブジェクトに関係するリンクパラレルファイル266はそのオブジェクトの二次データファイルに常駐する。

ユーザが所定のオブジェクトに対してオペレーションを行うよう選択すると、APPACK218のルーチンがそのオブジェクトのオブジェクトタイプ、オペレーションおよび可能なオブジェクトマネージャテーブルからの言語を読み取り呼び出すべき

対応のオブジェクトマネージャを検出する。特定のオブジェクトに対してオペレートするよう呼び出された特定のオブジェクトマネージャは実行すべきオペレーションのタイプにより変わり、かつあるオブジェクトマネージャは1タイプ以上のオブジェクトにオペレートすることができる。

オブジェクトプロトタイプテーブルはプロトタイプのオブジェクトを識別する。オブジェクトプロトタイプは対応するタイプのオブジェクトの基本的な省略値特性を含む。オブジェクトプロトタイプテーブルは、そのためにプロトタイプが存在するシステムに取り付けたオブジェクトの各タイプのプロトタイプコピイを識別する手段を提供する。そのためにプロトタイプの存在するいずれかのタイプの新しいオブジェクトが、オブジェクトプロトタイプのコピイを作ることにより作成され、プロトタイプオブジェクトのコピイはユーザにより修正されたり、オペレートされる新しいオブジェクトとなる。プロファイルエディタは新しいオブジェクトに対して対応する新しいプロファイ

ルを作成し、かつ必要に応じてオブジェクトの修正された特性を反映するために基本的プロファイルのコピイを修正するために使用しうる。

ユーザは、例えば書式文字であるデータをすでに含む新しいタイプのオブジェクトを作成できる。ユーザは前述のようにプロトタイプオブジェクトの適当なコピイをつくることにより新しいブランクオブジェクトを作成する。次いでユーザはプロトタイプオブジェクトに現われるべきデータを入力し、必要に応じてオブジェクトプロファイルを修正し、オブジェクトプロトタイプテーブルにこの新しいプロトタイプオブジェクトのエントリを位置する。次いで、この新しいプロトタイプオブジェクトは、プロトタイプオブジェクトのコピイを作ることによりブランクのプロトタイプオブジェクトと同様にオブジェクトを作成するために使用され、プロトタイプにユーザにより入力されたデータがプロトタイプの各コピイに現われる。

資源は、プログラムにより使用されるが、いずれのプログラムの実行可能コードの一部として記

(27) 憶されないデータのブロックである。これらの資源は、一般的に、例えば各種のオペレーティングシステムやユーザのインタフェース機能により本システムにより使用されるが、個々のプログラムにより使用される。１個以上のシステム資源を含む。さらに、資源は、特に本システムのユーザと特に関連し(ユーザファイルに記憶された)１個以上の資源を含みうる。

RESPACKは資源にアクセスしかつそれを修正するために使用するサービスや機能のパックである。いずれかのプログラムあるいはユーザを用いてRESPACKの装置を用いて資源を作成するか、あるいは既存資源をコピイし、かつコピイした資源をプログラム、機能またはユーザの特定の要件に合うようコピイした資源を修正しうる。これらの場合には、カストマイズされた資源がそのため作成あるいは修正されたプログラム、機能あるいはユーザに対して特定的となり、後述のように、プログラムあるいはユーザの関連ファイルの対応する修正によりそのプログラム機能あるいはユーザと

関連するようになる。その後カストマイズされた資源は元の未修正資源の代りに用いられる。

8.2.2. リンクテーブル

リンクテーブルはカタログのオブジェクトを参照したか、あるいは該オブジェクトによるレコードを含む。１個のカタログにおけるオブジェクト間のリンクに対して、２個のカタログの各々におけるリンクテーブルにはエントリがある。

2.2(a) 親のパーマネントＩＤ－12バイト

親のオブジェクトパーマネントＩＤ

2.2(b) 親の現在の位置－28バイト

オブジェクトの現在の位置はそのオブジェクトＩＤ(４バイト)、それがカタログ化されたボリュームの識別子(８バイト)、およびそのボリュームが位置しているシステムの識別子(16バイト)である。該システムの識別子は、それがこのレコードと同じシステムにあるときは零である。ボリュームの識別子はもしそれがこのレコードと同じボリュームにあれば零である。

2.2(c) リンクＩＤ－４バイト

親のオブジェクト内で独特のもの。リンクＩＤは親のオブジェクトのリンクマーカに記憶される。このリンクＩＤが特定のオブジェクトにおけるいずれか特定のリンクマーカに対応したリンクテーブルからのレコードの検索を可能とする。

2.2(d) リンクタイプ－１バイト

可能の値としては下記を含む。

子供:通常の値

バージョン:リンクされたオブジェクトは親のオブジェクトのバージョンである

反復:リンクされたオブジェクトは親のオブジェクトのコピイである

2.2(e) データリンク－１ビット

このリンクに対応するリンクパラレルアイルエントリにエントリがあるか否かセットされる。

2.2(f) 子供のパーマネントＩＤ－12バイト

子供のオブジェクトパーマネントＩＤ

2.2(g) 子供の現在の位置－28バイト

2.2(h) 子供のタイプ－４バイト

オブジェクトのタイア

2.2(i) 子供のタイトル－64バイト

性能を向上させるためにリンクレコードに保持された子供のオブジェクトのタイトルフィールドのコピイ

2.2(j) 子供の著者－32バイト

性能を向上させるためにリンクレコードに保持された子供のオブジェクトの著者のコピイ

2.2(k) コピイフラッグ－1ビット

もしセットされると、子供のオブジェクトは親のオブジェクトがコピイされるとコピイされる。もしセットされていないとすれば、子供のオブジェクトへのリンクがコピイされる。

2.2(l) 注釈－252バイト

リンクを記述するためにユーザが定義しうる自由テキスト

2.2(m) 削除フラッグ－1バイト

このリンクレコードが削除すべきかセットされる。この使用はデータベースシステムの内部へのものである。このため都合のよいときまでリンク

その元の名前空間(例えばファイル名あるいはデータベースの名前とレコード番号との組合せ)におけるオブジェクト部分の名前。

2.3(d) 遠隔フラッグ－1バイト

その部分がこのテーブルとは異なるシステムに位置するかセットされる。

8.2.4. フォルダテーブル

フォルダテーブルはオブジェクトテーブルにおける各フォルダ用のレコードを含む。

2.4(a) フォルダオブジェクトＩＤ－4バイト

フォルダオブジェクトのオブジェクトＩＤ

2.4(b) 作成ロケーション－64バイト

このフォルダにおいて作成されたオブジェクトを置くべき記憶位置

2.4(c) 作成ファイルクラス－1バイト

このフォルダにおいて作成された全てのオブジェクトのクラス

2.4(d) 作成ドキュメントライブラリ－1バイト

(28) レコードの物理的削除が延期される。

2.2(n) 同期 リンクフラッグ－1ビット

別のボリュームあるいはシステムに対してこのリンクレコードをそのコピイと同期すべきかセットされる。

8.2.3. ファイルリストテーブル

ファイルリストテーブルはその主要な部分以外のオブジェクトの各部分に対するレコードを含む。これらの部分は典型的にはファイルであるが、例えばレコードのような何か別のある。リンクパラレルファイルはこのテーブルにリストされる。

2.3(a) レコードの一連番号－4バイト

このファイルリストのレコードに対する独特の識別子

2.3(b) オブジェクトＩＤ－4バイト

このレコードにおいてそのオブジェクト部分(例えばファイル)が一部であるオブジェクトのオブジェクトレコードの識別子。

2.3(c) オブジェクト部分の位置－64バイト

このフォルダにおいて作成されたドキュメントオブジェクトを置くべきオブジェクトライブラリ

8.2.5. フィールドＷＩＴファイル

フィールドＷＩＴファイルはオブジェクトテーブルのＷＩＴ(「テキストのワード」)の索引付けされたフィールドにおける各ワードに対するレコードを含む。

8.2.6. オブジェクトＷＩＴファイル

オブジェクトＷＩＴファイルはＷＩＴの索引付けされたドキュメントの各ワードに対するレコードを含む。

8.2.7. 削除したオブジェクトテーブル

削除したオブジェクトテーブルはオブジェクトＷＩＴテーブルにおいて索引付けられそれ以来削除された各オブジェクト用のレコードを含む。

2.7(a) レコードの一連番号－4バイト

レコードの一意な識別子

2.7(b) オブジェクトＩＤ－4バイト

削除したドキュメントのオブジェクトテーブルのレコードを識別する。

8.3 リンクパラレルファイル

ボリュームは多くのリンクパラレルファイルを含むことができる。各プロジェクトに対して、少なくとも1個のデータリンク(リンク仕様を含むリンク)がはめ込まれたリンクパラレルファイルがある。リンク平行ファイルはデータ転送に使用可能のリンクに関する情報を含む。リンクパラレルファイルは前記リンクを含む各オブジェクトに対して存在する。

3(a) 更新状態−1バイト

リンクが更新されるときを制御する。以下の値を有しうる。

マニュアル:特にリクエストされると更新が行われる

ファーストタイム:親のオブジェクトの開放毎にせいぜい1回更新が行われ、かつ子供のオブジェクトが最初にアクセスされたとき更新が行われる

ダイナミック:子供のオブジェクトが変化する毎に更新が行われる

ことができる。

オートマチック:リンクされたデータがユーザ内に入れば常にリンクされたデータを表示する。

オンリクエスト:特にリクエストされたときのみリンクされたデータを表示する。さもなければ、リンクされたデータと親のデータとの関係に応じて、アイコンあるいはタイトルのみが表示されるか、あるいはオブジェクトが現われると(例えばドキュメントにイメージが現われると)空のスペースが表示されうる

3(d) リンク仕様ヘッダ

各リンク仕様に先行する標準情報は以下のフィールドを含む:

バージョン:8バイト

リンク仕様を作成した子供オブジェクトのオブジェクトマネージャのバージョンを識別する。このため、アプリケーションの改訂を促進し、一方リンク仕様をサポートする。

交換タイプ:4バイト

親オブジェクトに通す際それを通して子供のデータが交換されるフォーマット。

(29)

3(b) ディスプレイモード−1バイト

リンクを示すためにユーザが調べるものを制御する。

データ:リンクされたデータが表示される。

アイコン/タイトル:子供のオブジェクトのタイトルと、および/または子供のタイプのオブジェクトを示すアイコンが表示される。アイコンまたはタイトルの選択はアプリケーションに依存する。

3(c) ディスプレイ状態−1バイト

ディスプレイモードフィールドが、リンクされたデータがユーザに対して表示されるべきことを指示すると、ディスプレイ状態フィールドはリンクされたデータがユーザに対して表示されるときを制御する。表示しているリンクされたデータに関して多大のオーバヘッドがある可能性があり、例えばユーザは常にピクチャがドキュメントに現われるようにすることよりもドキュメントを速して迅速にスクロールできることを好む。このフィールドによりユーザがこの性能の選好を表現する

後続のリンク仕様情報のバイトの数

3(e) リンク仕様情報−可変長

子供オブジェクトのタイプに細部依存する。このフィールドはリンクにより参照される子供オブジェクト内のデータを識別するために(リンクがその中にはめ込まれているオブジェクトのオブジェクトマネージャあるいはアプリケーション統合サービスによるのでなく)子供のタイプのオブジェクトのオブジェクトマネージャにより解釈される。

3(f) リンクされたデータのコピイ−可変長

リンクされたデータのコピイを含む。

9. オブジェクトマネージャ呼び出し

9.1. カーネルを直接用いることによる呼び出し

オブジェクトマネージャは呼び出しプログラムの子供のオブジェクトのプロセスとして別のプログラムがオブジェクトマネージャを直接呼び出す

ことにより呼び出すことができる。しかしながら、オブジェクトマネージャの呼び出しは典型的にはAPPACKの呼び出し機能(例えばAPrqedit()、APinvoke())の１つを別のプログラムが呼び出すことにより開始される。

２種類のアプリケーションが非標準的なプロトコルを介してやりとりする必要のないときでさえ、カーネル呼び出しによる直接呼び出しに対して(専用リクエストを用いて)APPACK呼び出し機構の使用の選択の方が利点がある。例えばAPPACKはアプリケーション間のやりとりの設定により高レベルの援助を提供する。さらに、２種類の通信アプリケーションは相互から、双方の連続したオペレーションを独立して分離させることができる。これはAPPACK呼び出しが用いられると、双方のプロセスがアプリケーションマネージャプロセスの子供であり、２つの通信プロセスの一方が他方の子供とならないためである。

9.2. スタートアップリクエスト

本発明のシステムはオブジェクトに対して実行

でよいのでオプションのリクエストと称されている。

標準リクエストに加え専用リクエストをサポートするよういずれのアプリケーションも構成しうる。このため、必要とするどのような特殊のリクエストもどのアプリケーションも定義し、一方APPACKによりサポートされた基本的呼び出し機構を用いれるようにする。大量のオブジェクトタイプに対しても特定の専用リクエストは一般的にサポートされないので、典型的にはその他のアプリケーションがそのようなリクエストを出すことはない。それにもかかわらず、アプリケーションのある組合せは特別仕様のリクエストから著しい利点を得て、そのためそれら自身の特異なリクエストを共に定義し、サポートする。

基本的には、専用リクエストと標準リクエストとの間の差異はリクエストをサポートするアプリケーションの数である。特定のアプリケーションの構成者は専用リクエストの定義を広く公表するよう選択しうる。その他多くの者がリクエストに

(30) しうる多数の一般的オペレーションをサポートする。これらのオペレーションは「スタートアップリクエスト」と称される。その理由はまず、それらが実行すべきオペレーションに対するリクエストとして開始され、第２に各リクエストがオペレーションあるいはプロセスを開始させるからである。

本発明のシステムの重要な特徴は、概ねいずれのオペレーションも概ね全てのタイアのオブジェクトに対して実行しうるよう一組のオペレーションが定義されることである。本発明のシステムのオブジェクトマネージャは、実用上あるいは有意的に出来るだけ多くの標準的リクエストがオブジェクトの各タイアに対してサポートされるよう構成されている。ファイルベースのオブジェクトに対して、標準的リクエストのあるものはAPPACKにより直接実行しうる。これは、APPACKにより実行されるファイルベースのオペレーションが不十分である場合、オブジェクトのタイアに対するオブジェクトマネージャによりサポートされるのみ

対応するにはオブジェクトマネージャを修正するに十分有用なリクエストを見出すことができる。その結果、そのリクエストは事実上の標準リクエストとなる。

(APPACKにより典型的には直接実行される「オプションの」リクエストを含む)標準リクエストは以下の通りである：

START:省略オペレーションを実行する。多くのタイアのオブジェクトに対して、省略時オペレーションはユーザにより編集するようオブジェクトを開くことである。何らオブジェクトが規定されなければ、典型的な省略時オペレーションがユーザが新しいオブジェクトを作成できるようにする。

EDIT:ユーザがオブジェクトを編集できるようにする。

READ:ユーザがオブジェクトを観察できるが、修正はできないようにする。

RUN:規定したオブジェクトを実行する。オブジェクトマネージャは通訳あるいはプログラムロ

ーダとして機能する。

PRINT:プリントするようオブジェクトを整列させる。このリクエストにより規定されたオプションはユーザがプリントパラメータを提供するか、あるいは省略時パラメータを使用すべきかを指示する。

UPDATE LINK:特定のリンク仕様により指定されたデータの最終バージョンを親のオブジェクトに通信する。もしリンク仕様が有効でなければ、子供のオブジェクトマネージャがウインドウを開き、ユーザがリンクを再規定できるようにすべきである。

RESPECIFY LINK:ウインドウを開放し、ユーザがリンク仕様を変えれるようにする。子供のオブジェクトマネージャは(現在規定されているデータを示して)子供のオブジェクトの関連部分を表示し、かつユーザが種々データを規定できるようにする。

EDIT OBJECT PROFILE:ウインドウを開き、ユーザが子供のオブジェクトのプロファイルを編集

できるようにする。オブジェクトプロファイルに含まれている情報の性質はオブジェクトタイプによって変わる。

CUSTOMIZE:ユーザが、リクエストを受取るオブジェクトマネージャの挙動をカストマイズできるようにする。修正可能の特性が関連のデータを資源に記憶させている。カストマイズされた資源はこのオペレーションの結果である。ほとんどのプログラムはカストマイズ可能の挙動オプションを有しておらず、そのためこのリクエストをサポートしない。

CONVERT OUT:規定されたオブジェクトの中味全体を提供する。そのようなリクエストを受け取ると、オブジェクトマネージャは、(情報損失の増える順序で)提供しうる交換フォーマットのリストを含む、マッチメーカを備えたAPHNOPCONVERTOUTオペレーションを通知する。

CONVERT IN:新しいオブジェクト受取りデータを作成する。そのようなリクエストを受け取ると、オブジェクトマネージャは(情報損失の増え

る順序で)受け入れ可能交換フォーマットのリストを含む、マッチメーカを備えたAPHNOPCONVERTINオペレーションを通知すべきである。次いで、マッチメーカによりマッチメッセージが発行され、2個のオブジェクトマネージャ間の接続が作られデータを転送する。

COMPACT:規定されたオブジェクトにおけるデータ記憶をコンパクトにする。少なくともメッセージの完了、あるいはエラーメッセージの確認を含むブレティン(bulletin)をリクエスト元へ送る。

RECOVER:規定されたオブジェクト内のデータ記憶を回復する。メッセージの完了あるいはエラーメッセージの確認を含むブレティンをリクエスト元へ送る。

WORD LIST:オブジェクトにおける全てのワードのリストをコンパイルし、ブレティンをリクエスト元へ送る。ワードはバニラテキスト交換フォーマットを用いて戻される。それらの順番はフレーズオリエンティッドな索引付けのサポートに重要である。

CREATE:(オプション)
新しいオブジェクトを作成する。

COPY:(オプション)
既存オブジェクトのコピイを作成する。

RENAME:(オプション)
オブジェクトのユーザの可視の名前を変更する。

DELETE:(オプション)
オブジェクトを削除する。

FREEZE:(オプション)
オブジェクトを凍結する(例えば、オブジェクトテーブルのオブジェクトのレコードのフィールドを修正する)

INPORT:(オプション)
既存ファイルを取り上げオブジェクトとする。

9.8. APPACKによる呼び出し

APPACKによるオブジェクトマネージャの呼び出しを用いることに係るステップの順序は一般的に次の通りである。

(a) リクエスト元が、リンクを識別してAPPACK開始リクエストルーチンの1つを呼び出す(特に

汎用APinvoke()機能が用いられた場合より多くのパラメータが規定しうる)。

(b) APPACKルーチンがリクエストをリクエスト元のメイルボックス名と共にアプリケーションマネージャプロセスへ送る。

(c) アプリケーションマネージャは(オペレートすべきオブジェクトのタイプに基き)リクエストをハンドルする必要のあるオブジェクトマネージャはどれか規定し、かつどのリクエストを規定すべきかを決め、必要とされたオブジェクトマネージャの例を実行する新しいプロセスをつくる(これがこのオペレーションにおける「サーバ」である)。

(d) 新しく呼び出されたオブジェクトマネージャの初期設定の一部として、アプリケーションマネージャにそのリクエストと関連の情報とを依頼する。

(e) そのリクエストが受け入れ可能であれば、サーバは、オペレーションを進行すべきことを示して、(データは常にコンシューマに対するサー

(32) バの方向に流れる必要はないが、コンシューマとしても知られる)リクエスト元へ答えを送る。もしリクエスト元により規定されたり、あるいはリクエストの性質が絶対的なものであれば、リクエスト元とサーバとの間が接続される。

(f) もしデータに対してリクエストがなされたとすれば、データ交換が前記接続を通して実行される。アプリケーションマネージャはもはやインタラクションの一部ではなくなる。

(g) リクエスト元とサーバとがオペレーションを終了すると、それらは接続を停止する。

APPACKルーチンはリクエストの発信の細部を扱う。前記ルーチンは通信に使用されるメイルボックスを定義し、それ自身のメッセージを作り、特にリクエスト用ルーチン(例えばAPrqprint()、であって、APinvoke()ではない)への呼び出しの場合呼び出しを完了させるために(かつある場合には通信が何ら必要とされない場合はオペレーションを完了させるために)必要な事象の処理の全てを実行する。呼び出されたAPPACKスタートアップ

(startup)機能が復帰すると、オペレーションは中断するか、あるいは新しいオブジェクトマネージャが開始され満足のいく答えを出している。通信が要求されるとすれば、リクエスト元とサーバとは各々、さらに別の通信で用いる共通のオペレーションＩＤが提供される。オペレーションが終了すれば、双方がAPopfinish()を呼び出す。

標準的機能のいずれかをサポートするオブジェクトマネージャが開始され、(リクエストが何らユーザインタラクションを含まないこともあるので)ウインドウを開く前に、APint()を呼び出しスタートアップリクエストとパラメータとを取得すべきである。次いで、オブジェクトマネージャはリクエストを受けうるか決定すべきかであって、このことは典型的には、規定されたオブジェクトを開き、リンク仕様を有効にする(これはリクエスト元が依然として答えを待っているので、迅速にすべきである)ことを含む。もしリクエストを受けえないとすれば、オブジェクトマネージャは(リクエストを拒否する理由をつけて)APreply()

を呼び出し、かつクリヤにして出ていくべきである。もしリクエストを受けつけできるならば、このことがAPreply()を用いて報告されるべきである。リクエスト元との通信が必要とされれば、リクエスト元からのメッセージを待機し、その後オペレーション特有の通信が進行する。もしオペレーションがリクエスト元との通信を必要しないとすれば、オブジェクトマネージャはオペレーションを完了させるよう進行し、完了するとAPopfinish()を用いて状態を報告すべきである。

10. オブジェクトマネージャ(第３図)

特定オブジェクトにオペレートすべく呼び出された特定のオブジェクトマネージャはオブジェクトタイプによって変わり、かつさらに実行すべきオペレーションのタイプによって変わり、例えばワード処理のような言語依存型オペレーションにおいてはオペレートすべきオブジェクトで用いられる特定の言語によって変わりうる。

第３図を参照すれば、テーブルのデータフィールドのあるものを示すオブジェクトマネージャテ

ーブル256の概図が示されてい　。オブジェクトマネージャテーブル256は複数のオブジェクトマネージャテーブルのエントリ500から構成されている。各オブジェクトマネージャテーブルのエントリ500の方はオブジェクトタイプ、オペレーションのタイプ、即ち前記オブジェクトタイプに対して実行すべきリクエスト、適用可能なら言語および特定のリクエストを実行すべき対応するオブジェクトマネージャの識別との組合せに対応する。各オブジェクトマネージャテーブルのエントリ500はオブジェクトタイプフィールド502、リクエストフィールド504およびオブジェクトマネージャ識別子フィールド506とを含む。

前記フィールドの各々を検討すれば、オブジェクトタイプフィールド502はオブジェクトタイプコードを含む。リクエストフィールド504の方はオブジェクトマネージャテーブルエントリ500で識別されたタイプのオブジェクトに対して、あるいは関連して実行しうるオペレーションに対応するコードを含む。最後に、オブジェクトマネー

(33) ジャＩＤフィールド506は、特定のオブジェクトに対してリクエストされたオペレーションを実行するよう呼び出された特定のオブジェクトマネージャのファイル名を含む。

オブジェクトマネージャテーブルは以下のように使用しうる。ユーザに対してオブジェクトを表示するためにオブジェクト１つにおいてデータを読取る過程においてオブジェクトマネージャはリンクマーカを見い出しうる。リンクマーカはリンクＩＤを含む。このリンクＩＤはリンクマーカがはめ込まれたオブジェクト(このリンクに関する「親の」オブジェクト)のパーマネントＩＤと共にリンクテーブルにおいてそのリンクと対応するレコードを探索するために使用しうる。このリンクレコードはリンクされたオブジェクトのタイプを識別するフィールドを含む(オブジェクトテーブルへのキーとしてリンクレコードにおいて子供のパーマネントＩＤを用いることにより子供のオブジェクトのオブジェクトタイプを取得しうるので、このフィールドは必要とされない)。子供のオブ

ジェクトに対するこのタイプコードは「ディスプレイ」のリクエストのコードと共にオブジェクトマネージャテーブルにおけるレコードを識別するために使用される。そのレコードのオブジェクトマネージャＩＤはリンクされたオブジェクトに対して必要とされるディスプレイオペレーションを実行しうるプログラム用のファイル名である。

11. オブジェクトプロトタイプテーブル(第４図)

第４図を参照すれば、オブジェクトプロトタイプテーブル258は本システムに取り付けられたオブジェクトの各タイプの記憶したプロトタイプコピイをアクセスする手段を提供する。第４図に示すように、各プロトタイプテーブル258は複数のオブジェクトプロトタイプエントリ600より構成され、各オブジェクトプロトタイプエントリ600は単一のユニークなオブジェクトプロトタイプに対応する。

各オブジェクトプロトタイプエントリ600はオブジェクトタイプフィールド602、言語フィールド604およびプロトタイプ識別子フィールド606から

ら構成されている。オブジェクトタイプフィールド602と言語フィールド604とがオブジェクトプロトタイプエントリ600をアクセスする「キー」を構成し、一方プロトタイプＩＤフィールド606は、各キーに対して、対応するプロトタイプオブジェクトを識別するためにオブジェクトプロトタイプエントリ600から抽出した情報を含む。前記フィールドの各々を検討すれば、オブジェクトタイプフィールド602は、そのプロトタイプがオブジェクトプロトタイプエントリ600により識別されるオブジェクトのタイプの識別子を含む。言語フィールド604の方は、そのプロトタイプがオブジェクトプロトタイプエントリ600により識別されるオブジェクトのタイプの特定バージョンが言語依存型である場合の特定の言語を識別するコードを含む。例えばドキュメントオブジェクト、およびしたがってそれらのプロトタイプが、第１のタイプが英語のドキュメント、第２のタイプが独語のドキュメントおよび第３のタイプが仏語のドキュメントというように区別される数種のドキュメン

トタイプオブジェクトのプロトタイプがありうる。最後に、プロトタイプＩＤフィールド606は、そのタイプのオブジェクトのプロトタイプコピイを含むファイルのファイル名を含む。

ユーザは例えばメニューピック(menu pick)を介してオブジェクトタイプと、および可能ならオブジェクトの言語バージョンを識別することにより所定タイプの新しいオブジェクトを作成しうる。次いでオブジェクトタイプ識別子と言語コードとがオブジェクトプロトタイプテーブル258で対応するオブジェクトプロトタイプエントリ600を探すキーとして用いられ、対応するプロトタイプオブジェクトを含むファイルの名前がオブジェクトプロトタイプエントリ600から読みとられる。次いでファイルコードユーティリティを呼び出すためにプロトタイプファイル名が使用され、前記ユーティリティの方はプロトタイプの新しいコピイを作る。新しいオブジェクトに対するエントリが次いでオブジェクトテーブルにおいて作成される。

ユーザは次いで、そのオブジェクトタイプのオ

ブジェクトマネージャを呼び出し、そのオブジェクトに対してオペレートするかあるいはそのオブジェクトタイプのプロファイルエディタを呼び出し、必要に応じ、あるいは希望に応じ新しく作成されたオブジェクトをカストマイズできる。もしユーザが新しいオブジェクトタイプの作成を希望していたとすれば、ユーザは適当なプロファイルエディタを用いて必要に応じ、あるいは希望に応じ新しく作成されたオブジェクトのプロファイルをカストマイズし、あるいはプロトタイプで所望されているデータを入力しうる。次いで、ユーザはユーティリティを呼び出して新しいオブジェクトタイプに対応する新しいオブジェクトプロトタイプエントリ600を作成し、かつこの新しいオブジェクトプロトタイプエントリ600をオブジェクトプロトタイプテーブル258に位置させる。次いで、ユーザは新しいタイプの新しいオブジェクトを前述のオペレーションを介して意のままに作成しうる。

12、オブジェクトカタログ(第５図、第６図および第７図)

第５図を参照すれば、オブジェクトカタログ252の構造と構成とが示されている。オブジェクトカタログ252はオブジェクトテーブル260、リンクテーブル262およびファイルリストテーブル264とを含むものとして示されている。これらはオブジェクトと、オブジェクトへのリンクについての情報を含む。

第６図を参照すれば、オブジェクトテーブル260は１個以上のオブジェクトレコード360から構成されている。オブジェクトカタログ252に含まれた各オブジェクトに対して１個のオブジェクトレコード360がある。各オブジェクトレコード360は、オブジェクト識別子362、オブジェクトタイアコード364およびオブジェクト位置の指示366とを含む複数のフィールドに構成された情報を含む。

第７図を参照すれば、リンクテーブル262は１個以上のリンクレコード370から構成されている。オブジェクトテーブル260でカタログ化されたオブジェクトへ、あるいはそこからの各リンクに対して１個のリンクレコード370がある。各リンクレコード370は、親のオブジェクト識別子372、リンク識別子374、子供のオブジェクト識別子376、リンクタイアコード378、データリンクフラッグ380およびコピイフラッグ382とを含む複数のフィールドに構成された情報を含む。また、子供のオブジェクトの現在の位置もリンクテーブル262に記憶しうる。

コピイフラッグフィールド382は親のオブジェクトの識別子372によって識別されたオブジェクトをコピイするために使用される。親のオブジェクト400のコピイが作られると、リンクを親のオブジェクトと共にコピイするか、あるいは子供のオブジェクト識別子376により識別されたオブジェクトのリンクされたデータをコピイする必要がある。リンクまたは、リンクにより参照されるデータがコピイされるか否かはコピイフラッグフィールド382の状態によって決まる。

本発明のシステムにおいて、採用された規則は、もしリンクされたオブジェクトが親のオブジェク

トの内部から作成されたとすれば、子供のオブジェクト即ちそこからリンクされたデータはコピイされるか、あるいはリンクがコピイされるということである。例えばもしユーザがチャート(図表)を用い、そのためドキュメントタイプの親のオブジェクト内からの、あるいはその部分としてのチャートタイプの子供のオブジェクトは親のオブジェクトと重複するとすれば、親のオブジェクトのコピイはチャートの新しいコピイを含む。チャートのこのコピイは親のオブジェクトのコピイと同様、原始バージョンと独立して修正しうる。しかしながらもしリンクが以前に存在していた親のオブジェクトと以前存在していた子供のオブジェクトとの間で作成されたとすれば、子供のオブジェクトのリンクされたデータよりもむしろリンクの方がコピイされる。

12.1. カタログサーバプロセス

本発明の好適実施例においては、オブジェクトカタログはホストコンピュータあるいはファイルサーバ(例えば第１Ａ図、第１Ｂ図および第１Ｃ

を使用することにより機密性を向上させることができる。

13.リンクおよびリンクパラレルファイル(第８図)

前述のように、本システムにおける全ての情報、データおよびプログラムはオブジェクト内に含まれ、オブジェクトの方はリンクを介して相互に関連する。リンクは、「子供」のオブジェクトと称される一方のオブジェクトが「親」のオブジェクトと称される他方のオブジェクトに接続される手段と見做すことができる。各オブジェクトは典型的には少なくとも１個のリンクに対して子供である。例えば、各ユーザは少なくとも１個の一次フォルダオブジェクト即ちフォルダを有し、該フォルダにはユーザが直接あるいは間接的にリンクされる。いずれの数のオブジェクト、あるいはオブジェクトの部分もリンクを介して相互に結ぐことができ、リンクの順序および方向は階層的に限定されない。

子供のオブジェクトを親のオブジェクトに結合することの他に、リンクはまた、子供のオブジェクトのデータの全て、あるいは一部を親のオブ

(35) 図のそれぞれ170,116または156)に対して実行するカタログサーバ190を用いて実施される。前記ホストコンピュータは中央の大容量メモリ(例えば、172,110,114)を支配する。代案は記憶されたカタログデータへの直接アクセスをカタログからの情報を必要とする(あるいはカタログに情報を追加する)各プロセスが出来るようにさせることである。

カタログサーバを用いることの利点のあるものは以下の通りである。カタログサーバはカタログファイルを常に開放しうるよう連続的に使用されるプロセスであって、カタログサーバの使用によりカタログにアクセスする各々の個別プロセスに対してカタログを開けておく必要性を排除する。同様に、カタログサーバは、さもなければさらに実行が困難となるバッファリングを提供しうる。カタログへのアクセスを必要として個々のプロセス全部が記憶されたカタログデータへの直接アクセスを提供されたとすれば、カタログの効果的な機密性を実行することが困難で、カタログサーバ

ジェクトに結合するためにも使用しうる。子供のオブジェクトから親のオブジェクトへのデータの結合は、結合されるデータが親のオブジェクトの一体部分となるのではなく、子供のオブジェクトの一部のままであるという点において一方のオブジェクトから他方のオブジェクトへデータをコピイすることは区別される。後述のように、リンクされたデータは子供のオブジェクト内のみで、子供のオブジェクトタイプに指定したオブジェクトマネージャによってのみ編集しうる。

さて第８図を参照すれば、リンクパラレルファイル265と、子供のオブジェクト404からの(子供のデータ406の)リンクされたデータ407が親のオブジェクト400に統合されている親のオブジェクト400と子供のオブジェクト404との間のリンクとが示されている。第８図に示され、かつ以下説明するリンク構造とリンクパラレルファイル265とはまた、リンクパラレルファイルが含まれていないことおよび子供のオブジェクトに何らリンクされたデータの無いことを除いて(例えばオブジェ

クトをフォルダタイプオブジェクトに結合する場合)非データリンクのリンクにも適用されることに注目すべきである。

第８図に示すように、リンクの主要エレメントとしては、親のデータ402を含む親のオブジェクト、子供のデータ406を含む子供のオブジェクト404および少なくとも１個のリンクパラレルファイルのエントリ408を含むリンクパラレルファイル266とを含む。

親のオブジェクト400にリンクすべき子供のデータ408の部分はリンクされたデータ407として子供のデータ406において示され、このリンクされたデータ407が親のデータ402にリンクすべき位置は親のオブジェクト400におけるリンクマーカ420の１つであるリンクマーカ420aによって示されている。

まず親のオブジェクト400、特に前述のリンクマーカ420aを参照すれば、各リンクマスタは対応する他のオブジェクト、あるいは別のオブジェクトのデータがそこを通して例えば親のオブジェク

(36) ト400のような親のオブジェクトにリンクされる関連リンクへの基準である。各リンクマーカは、親オブジェクト内の対応するリンクを独特の方法で識別するリンクＩＤを含む。さらに、各リンクマーカは、リンクされた子供のオブジェクトあるいは子供のオブジェクトからのリンクされたデータが「現われる」親オブジェクトでの個所をマークする。

親オブジェクト400のリンクパラレルファイル266は親オブジェクト400にはめ込まれた各データリンク用のリンクパラレルファイルエントリ408を含む。これらリンクパラレルファイルエントリ408の各々は、親オブジェクト400に現われるべき子供オブジェクトのデータを識別する情報を含む。

リンクプロフィルフィールド422はリンクとリンクされたデータの状態に関するある基本的情報を提供する。特に、リンクプロフィルフィールド422は、フィールドの更新状態、ディスプレイモードおよびディスプレイ状態とを含む。親オブジェクト400におけるリンクとリンクされたデータと

の外観と作用とに関する情報を含む。

更新状態フィールドの値は、いつ対応するリンク266が更新されるか、即ち子供オブジェクト404の新しい現在のコピイ並びに子供オブジェクト404からのリンクされたデータが取得され、親オブジェクト400に提供されるかを決める。リンクのこの更新は自動的に、即ちユーザが何ら干渉することなく行われ、ある事象が発生すると、即ちユーザがリクエストしたときのみマニュアルとなるか、あるいはそれらの組合せとなる。

自動更新に対して更新状態フィールドが設定されると、親オブジェクト400のオブジェクトマネージャは、親オブジェクトのオブジェクトマネージャが決めるときに、子供オブジェクト404のコピイである子供オブジェクトのデータを自動的に取得し、一時的に記憶させることができる。このことは例えば、リンクされたデータ407を表示あるいはプリントする毎、あるいは親オブジェクト400が開始する毎、即ち親オブジェクト400に対するオペレーションの開始時に発生する。また、本

システムはダイナミック更新即ち、子供オブジェクト404に対するオペレーションにより更新が開始され、そのため子供オブジェクト404における変化が、子供オブジェクト404が変ると親オブジェクト400に自動的に現われる。しかしながら、マニュアル更新にマニュアル状態フィールドが設定されると、リンクはユーザにより特別のリクエストあるいは指令のときのみ更新される。

ディスプレイモードフィールドはセット可能のフィールドを含み、該フィールドは対応するリンクを親のオブジェクト400で表示する態様を選択できるようにする。オブジェクトのタイトル、オブジェクトのタイプを示すアイコン、子供オブジェクトのデータを示すことにより、あるいはこれらをいずれか組合せることによりリンクを示すことができる。

ディスプレイ状態フィールドをセッティングすることにより、親オブジェクト400でデータ用にとっておいた領域が例えばＣＲＴディスプレイのような可視表示でユーザに見えると、下記するリ

ンク仕様に含まれているデータが自動的に表示されるか否か検出する。ディスプレイ状態フィールドが「自動ディスプレイ」にセットされると、リンクされた子供オブジェクト404のデータが、親オブジェクト400の領域がユーザに見えてくると常に表示される。ディスプレイ状態フィールドが「オン・コマンド(指令)ディスプレイ」にセットされると、リンクされた子供オブジェクト404のデータが、ユーザによるディスプレイに対する特別の指令が出るときのみディスプレイされる。もしユーザが「オン・コマンド・ディスプレイ」モードにある間にリンクされたデータをディスプレイせよとの指令を出さないとすれば、リンクは例えばリンクされた子供オブジェクト404のデータにとっておいた領域の外形と、例えばアイコンあるいはリンクのタイトルのようなリンクされた領域の短い指示とにより表示される。ディスプレイ状態フィールドのセッティングと作用とは、もしディスプレイモードフィールドがリンクされたデータを表示しないよう、即ちリンクをタイトルあるい

いずれの子供オブジェクトのデータが親オブジェクト400にどのような形態で提供されるか検出するために子供オブジェクトのオブジェクトマネージャにより用いられる。この目的に対して、リンク仕様フィールド438に常駐する情報が、前述のようにリンクが更新される毎にAPPACK218更新ルーチンにより子供オブジェクトのオブジェクトマネージャに戻される。子供オブジェクトのオブジェクトマネージャが次いでこの情報を解釈して、リンク仕様によりどの子供オブジェクト404のデータが示されるか、どのような形態でリンクされたデータが親オブジェクト400に提供されるか検出し、かつ期待した形態でリンクされたデータを親オブジェクト400へ提供する。

リンク仕様フィールド438を参照すれば、これはそれぞれヘッダフィールドとリンク仕様情報フィールドと称される２ブロックのフィールドから構成されている。ヘッダフィールドは主として子供オブジェクト404の記述する情報を含み、一方リンク仕様情報フィールドは子供オブジェクトのリ

(37) はアイコン、あるいはタイトルとアイコンの組合せによってのみ表示するようセットされると適用できないことに注目すべきである。

リンク仕様フィールド438は、リンクプロファイルフィールド422で述べたように親オブジェクトのリンクの表示と作用よりもむしろ子供オブジェクト404からのリンクされたデータを記述する情報を含む。

リンク仕様フィールド422に含まれた情報は、子供オブジェクト404からのリンクされたデータ(リンクされたデータ407)が親オブジェクトで表示され、かつ使用される態様を決めるために、親オブジェクト400のオブジェクトマネージャでなく、むしろ親オブジェクトにより使用される。対照的に、以下説明するように、リンク仕様フィールド438の情報がリンクパラレルファイルのエントリ408で提供され、そのため親オブジェクト400と関連するが、その中の情報は親オブジェクトのオブジェクトマネージャにより使用されない。その代りに、リンク仕様フィールド438の情報は、

ンクされたデータを記述する情報を含む。

第６図に示すように、まずヘッダフィールドを参照すれば、該ヘッダフィールドはバージョンのフィールドと、長さのフィールドと、および交換タイプのフィールドとを含む。バージョンフィールドは元々リンク仕様を作成した子供のオブジェクトマネージャの特定バージョンを識別する。長さのフィールドはリンク仕様データの長さを指示する。交換タイプフィールドは子供オブジェクトから親オブジェクトまで、リンクされたデータをやりとりする上で使用すべきデータ交換フォーマットを示す。

交換タイプフィールドに関して、オブジェクトの各タイプは異なるタイプあるいは異なるデータのフォーマットを有することがある。したがって、リンクの子供オブジェクト404と親オブジェクト400とは異なった形態のデータを有することが可能である。例えば、ピクチャ、チャートあるいは図面のようなピクチャあるいはグラフィック(図形)データはグラフィックタイプの子供のオブジェク

トからドキュメントタイプの親のオブジェクトへリンクできる。多くの場合、リンクされたデータは親オブジェクトにおけるものと同じタイプのものでよく、あるいはデータが子供データタイプから、情報を僅か、あるいは全く損失することなく親のデータタイプまで部分的あるいは完全に変換しうるに十分類似でよい。しかしながら、その他の場合、リンクされたデータは、親オブジェクトで直接使用可能のタイプに容易には変換しえない。

オブジェクトの間でのデータの変換は、原始オブジェクト即ち子供オブジェクトのデータのタイプと宛て先の、即ち親のオブジェクトのデータのタイプとによって左右される。もしデータタイプが同一あるいは十分近似であるならば、データは直接変換され、データをほとんど、あるいは全く変換せず、即ち情報をほとんど、あるいは全く損失することなく宛て先オブジェクトにおいて直接使用することができる。しかしながら、もしデータタイプが十分異なっているとすれば変換は困難であり、あるいはデータは宛て先オブジェクトに

(38) おいて直接即ち宛て先オブジェクトのデータの一体部分として使用することができない。この場合、ある程情報損失があり、かつ後述するように原始オブジェクトのデータは出来るだけ良く、宛て先オブジェクトのデータに「はめ込む」必要がある。

オブジェクト間のデータ交換は「マッチメーカ」と称するAPPACKファシリティの援助を得て達成することができる。マッチメーカはデータ交換オペレーションの２個所の終りに関連した２個のオブジェクトマネージャが相互に識別し合い、かつ共通のデータ交換フォーマットを識別しやすくさせるために使用される。

さて、リンク仕様情報フィールドを参照すれば、これらのフィールドは前述のように子供オブジェクト404からリンクすべきデータを記載している。リンク情報フィールドの特定の構造はオブジェクトのタイプによって左右される。一方のフォーマットにおいて、リンク情報フィールドは、例えば子供データ406における リンクされたデータ407の始めを指示し、かつリンクされたデータ407内

に含まれた子供データ406の部分と識別するデータ識別フィールドとを指示することにより子供データ406でのリンクされたデータ407の位置を指示する位置フィールドを含む。

子供データ406は、リンクされたデータ407が親オブジェクト400にリンクされた後で編集、あるいは変更または修正が可能であり、かつ子供オブジェクト404の変更はリンクされたデータ407の位置や識別に影響を与えうることを注目すべきである。したがって、リンクされたデータ407の位置や識別が指示される態様、即ち位置フィールドやデータ識別フィールドに位置すべく選択した情報のタイプは出来るだけ子供データ406に対する変更に敏感でないことが必要である。リンクされたデータ407が位置され、かつ識別される態様はリンクされたデータ407あるいはリンクされたデータ407を囲む子供のデータ406がリンク仕様を無効にすることなく変化できるようにするか、あるいは子供オブジェクト406のオブジェクトマネージャが変更のなされた後リンクされたデータ407の位

置や識別の再設定できるようにさせる。例えば、位置フィールドとデータ識別フィールドとに記憶されている情報は、子供データ406の領域に対するリンクされたデータ407の識別された領域を位置づけることができる。さらに、データ識別はリンクされたデータ407でのカーソルの初期位置を含むことを注目すべきである。

最後に、リンクパラレルファイルのエントリ408の最後のフィールドを参照すれば、リンクパラレルファイルエントリ408はリンクされたデータのコピイフィールド458を含むことが示されている。前述のように、子供オブジェクト404からのリンクされたデータは、アイコン、タイトルあるいはリンクされたデータ自体、あるいはこれらのいずれかの組合せにより親オブジェクト400においてユーザに表示されうる。リンクされたデータコピイのフィールド458は、リンクされたデータがアイコンあるいはタイトルとの組合せ如何を問わずデータ自体により可視表示される場合に、子供オブジェクト104からのリンクされたデータのコピ

イを記　する手段である。リンクされたデータがユーザに可視表示される際表示されるのはリンクされたデータコピイのフィールド458に記憶されたリンクされたデータのコピイである。リンクされたコピイがアイコンあるいはタイトルのみにより表示されると、リンクされたデータコピイのフィールド458は何らデータを含む必要はない。

リンクが形成され、リンクが更新される毎にリンクされたデータの新しい現在のコピイが子供のオブジェクト404から受取られ、リンクされたデータコピイのフィールド458に記憶される。

第８図にはまた、リンクテーブル262とオブジェクトテーブル260とのリンクに対する役割も示されている。リンクマーカ420aと親オブジェクト400のＩＤとがリンクテーブル262のエントリを探すために使用される。このエントリは、子供オブジェクト404に対応するオブジェクトテーブルにおけるエントリを探すために使用する子供識別情報を含む。オブジェクトテーブル260におけるこのエントリは子供オブジェクト404を探すために使用

(39) する情報を含む。

14. コピイ、移動および共用

本発明によるシステムにおいて実施可能のコピイ、移動および共用オペレーションについての以下の説明は、先に説明してきたこれらオペレーションの局面を要約し、さらにこれらオペレーションの別の局面を説明するものである。前記オペレーションについての以下の説明は、種々オブジェクトの間でのコピイ、移動および共用オペレーションに特に関するものであることを注目すべきである。これらのオペレーションは単一のオブジェクト内で実施可能であるが、共用オペレーションを除いて、あるオブジェクト内でオペレーションが実施される正確な方法と手段とは、関連のオブジェクトマネージャにより決定される。しかしながら、オブジェクト内のデータの共用は、全てのオブジェクトマネージャがオブジェクト間の共用をサポートする訳ではないが、オブジェクト間の共用と同じ要領で、かつ同じ手段により実行される。

本発明によるシステムにおけるコピイ、移動お

よび共用オペレーションは本システムのユーザにとっては、原始オブジェクトからのデータのコピイが各オペレーションにおいて宛て先オブジェクトに現われる点で基本的に類似に見えるであろう。コピイと移動オペレーションは、原始データのコピイが宛て先オブジェクト「内に位置され」かつ下記するある例外内で、宛て先オブジェクトの一部となる点で類似である。コピイと移動オペレーションとの主たる相違は、移動オペレーションにおいて、原始データは原始オブジェクトから削除され、一方コピイオペレーションにおいて原始データが原始オブジェクトにおいて不変のまま残る。コピイと移動オペレーションとは以下説明するように、コピイされたデータあるいは移動されたデータが宛て先オブジェクトにおいて、該宛て先オブジェクトのオブジェクトマネージャがコピイされ、あるいは移動されたデータのタイプに対してオペレート出来る程度まで宛て先オブジェクトにおいてオペレートできるという点でさらに類似である。

一方のオブジェクトから別のオブジェクトへの

データの共用は、共用されたデータが親のオブジェクトの一体部分となるのでなく子供オブジェクトの一部として留まるという点で一方のオブジェクトから別のオブジェクトへのデータのコピイや移動とは区別される。共用されたデータはリンクにより親のオブジェクトで利用可能とされる。共用されたデータは子供オブジェクト内でのみ、かつ子供オブジェクトタイプに指定されたオブジェクトマネージャのみによって編集可能である。

共用は、リンクが「更新」されうるためデータのコピイあるいは移動とさらに区別される。コピイあるいは移動オペレーションに対してそのような進行中のキャラクタはない。共用オペレーションから生じるデータは後のリンク更新オペレーションにより変化しうる。宛て先オブジェクトをオペレートする毎、例えば開放したり、表示したり、編集したり、あるいはプリントする毎に更新するようリンクをセットしうる。あるいは原始オブジェクトが修正されれば常に更新するようリンクをセットしてもよい。あるいは、リンクはマニュ

アルで更新するようセットしてもよく、この場合共用されたデータは更新が特にリクエストされるまで効果的に「凍結」される。

コピイと移動とは、コピイあるいは移動されたデータを含む宛て先オブジェクトの部分に対してコピイあるいは移動オペレーションが実行されると、宛て先オブジェクトに常駐しているコピイあるいは移動されたデータが処置されるという点において共用とさらに相違する。即ち、コピイあるいは移動されたデータはデータタイプに関し以下さらに説明する限度内で宛て先オブジェクトの一部として処理され、宛て先オブジェクトにおける他の原始部分と同じ要領でコピイあるいは移動しうる。この点に関して、データ交換に関し前述したように、データ交換フォーマットはデータ交換のデータ流れの間データタイプを変えるための配備はしてあることに注目すべきである。即ち、宛て先オブジェクトのそれとは異なるタイプの先にコピイあるいは移動したデータは、コピイあるいは移動されつつある宛て先オブジェクトの部分に

(40) はめ込まれているのが判る。そのような場合に、前述のように本発明のシステムにおけるデータストリームの転送に採用した規則では、マッチメーカ810のオペレーションにより定義されるデータストリーム用の初期交換タイプあるいはデータフォーマットがあることである。別のデータタイプが例えばリンクマーカあるいはリンクされたデータのようなストリームへ挿入されるとすれば、データタイプの変更と新しいフォーマットとを示す指示コードが前記ストリームの中へ挿入され、適当なデータフォーマットの交換により呼び出しを促進し、データ転送／変換ルーチン808を新しいデータタイプに切り換える。例えば、データストリームの挿入された部分の終りにおいてデータタイプが再び変わると同じオペレーションが用いられ、交換フォーマットは原始データタイプに戻る。

対照的に、リンクは効果的にデータのコピイを原始オブジェクトから宛て先オブジェクトまで転送するが、リンクされたデータは実際には宛て先オブジェクトの一体部分ではない。例えば、リン

クされたデータは宛て先オブジェクトのオブジェクトマネージャによりオペレートはされえぬが、原始オブジェクトマネージャにより原始オブジェクトにおいてオペレートされねばならない。

別のオブジェクトからリンクされたデータを含むオブジェクトの一部をコピイあるいは移動させるべき場合、リンクされたデータあるいは実際のリンクされたデータへのリンクがコピイあるいは移動されるか否かは、リンクをはめ込んだオブジェクトがどのようにして作成されたかにより決定される。本発明によるシステムにおいては、採用された規則は、もし原始オブジェクトが別のオブジェクト内から作成された場合、リンクされたデータはコピイされるということである。例えば、もしユーザがドキュメントタイプのオブジェクトのドキュメント内から、かつその一部としてチャートを、したがってチャートタイプの原始オブジェクトを作成し、次いでドキュメントオブジェクトあるいはリンクされたチャートタイプデータを含むドキュメントオブジェクトの部分を複製し

てドキュメントオブジェクトのこの部分をコピイしたとすれば、ドキュメントオブジェクトのそのコピイはチャートの新しいコピイへのリンクを含む。このチャートのコピイは、ドキュメントのオブジェクトのコピイと同様原始バージョンとは独立して修正できる。しかしながら、もしリンクが先に存在していたドキュメントオブジェクトと先に存在していたチャートオブジェクトの間で作成されたとすれば、リンクされたデータよりもむしろリンクがコピイされ、その結果第１のドキュメントと同じチャートオブジェクトのリンクを備えたドキュメントのコピイを作ることになる(即ち前記チャートに対する変更が双方のドキュメントに現われることになる)。

原始データが宛て先データと類似かあるいは同一の場合、直接あるいは変換後、コピイされ、移動され、あるいは共用された原始データは宛て先データに「混入されるか」あるいは「はめ込まれる」ようになる。コピイあるいは移動されたデータの場合、コピイあるいは移動されたデータは宛て先

オブジェクトマネージャにより編集可能である。しかしながら、共用されたデータの場合、かつ共用されたデータがコピイ、移動あるいは削除される場合、該データは原始オブジェクトにおいてのみ、かつ原始オブジェクトのオブジェクトマネージャによってのみ編集可能である。宛て先オブジェクトのオブジェクトマネージャは共用されたデータを独自の内部データフォーマットに変換可能であるが、共用されたデータを独自のデータに組み入れることはなく、いずれにしても原始データを編集できず、共用データを「マークし保持する」必要がある。ユーザがデータが編集されつつあるオブジェクトの直接その一部でなく、共用されているのであることに気が付くようデータをユーザに対して可視マークを付けるべきである。「マークされ、保持されている」データの場合、宛て先オブジェクトのリンク平行ファイルのエントリはリンクされたデータのコピイを含む(即ち「保持して」いる)。リンクパラレルファイルにおけるこのコピイは直接編集でき、むしろこのデータはそ

(41) こからリンクされている原始オブジェクトを編集することにより編集される。

宛て先オブジェクトマネージャは、コピイあるいは移動オペレーションを首尾よく実行するためにコピイされ、あるいは移動されたデータを編集できる必要はなく、単に原始データを何らかの方法で宛て先オブジェクトへはめ込めればよい。コピイあるいは移動したデータを「密閉化」でき、その場合データは適当なタイプの個別のオブジェクトに記憶され、該オブジェクトへのリンクは、コピイあるいは移動オペレーションの終りにあるオブジェクトにはめ込まれる。

共用オペレーションにおいては、宛て先オブジェクトマネージャは共用オペレーションを定義することにより何らかの方法で原始データを編集し、共用されたデータを「マークし保持する」必要がある。共用されたデータはコピイ、移動あるいは削除しうるが、一方編集できるのは原始オブジェクトにおいてのみ、かつ原始オブジェクトのオブジェクトマネージャによってのみである。

15. マッチメーカ

15.1. マッチメーカの目的と全体的作用

２個のオブジェクトの間でデータ転送あるいはリンクを発生させるには、ユーザは原始オブジェクトと宛て先オブジェクト並びに実行すべきオペレーションとを識別する。例えば、ユーザは原始データを選択し、COPYキーを押し、次いで宛て先を指し、PLACEキーを押す。原始とデータとが異なるオブジェクトにある場合、(異なるプロセスでそれぞれ実行している)２個のオブジェクトマネージャが介入させられる。しかしながら、あるプロセスが所有するウインドウの外側でのカーソルの移動はそのプロセスには知られていないので、プロセスはオペレーションの他の半分(例えばCOPYあるいはPLACE)がユーザにより実行されたことも知らない。この問題はマッチメーカにより解決される。

アプリケーションマネージャは、各々の潜在的なオペレーションに対して２側面を有するマッチメーカを保持する。即ち、サーバ側(即ち何かを

有する側)とコンシューマ側(即ち何かを欲している側)である。データ交換オペレーションに対しては、サーバ側(例えば、COPY、MOVE、SHARE)を通知する。オブジェクトマネージャはデータ交換フォーマットが提供するものを列挙する。コンシューマ側(例えばPLACE)を通知するオブジェクトマネージャはそれが受け入れるデータ交換フォーマットを列挙する。

一旦COPY、MOVE、またはSHAREが通知されると、PLACEが通知されるまで、それ以外のCOPY、MOVEまたはSHAREは受け入れられない。COPY、MOVEまたはSHAREが何ら通知されおらず、PLACEが通知されたとすれば、COPY、MOVEまたはSHAREが受け入れられるまで、(あるいは通知したオペレーションを担当するオブジェクトマネージャがオペレーションを停止するまで、)それ以上のPLACEは通知されない。

その他の明白に区別のつくオペレーションの対が定義されるとすれば、これらオペレーションのマッチメーカの半分が同時に通知され、満足のい

く半分が呼び出されると突き合わせ(マッチ(match))が行われる。

マッチングが行われ、(オペレーションのマッチングした側が通知されると)APPACKマッチメッセージがサーバとコンシューマの双方へ送られる。次いで各々がAPmm connect()を呼び出し相互に通信し、直接データを交換するに必要な情報を取得する。次いで、アプリケーションマネージャがマッチメーカをクリヤし、最初のマッチから発生したデータ交換が進んでいる個別のマッチを受け入れる。

15.2. マッチメーカプロトコル

15.2.1. 開始プロトコル

オブジェクトマネージャがCOPY、MOVEあるいはSHAREイベントを受けとると、オブジェクトマネージャは次のステップをとる。

(a) オペレーションを通知する。

− APmmreserve()を呼び出し、同じタイプのオペレーションがすでに進行しているか検出する。もしその通りであれば、エラーメッセージを

いる)、(「何処へ？」"Towhere？"の処理をスキップして)オペレーションを開始する。

− もしAPmmpost()がまだマッチのなされていないことを示せば、「何処へ？」の処理を行う。

(b) 宛て先を決める(「何処へ？」)

− 「何処へ？」を表示する。

− もしPLACE事象が(このオブジェクトマネージャにより)受け取られるとすれば、オペレーションは内部的(即ち単一のオブジェクト内の一方の場所から他の場所)である。APmmclear()を呼び出しマッチメーカをクリヤにしオペレーションを進行させる。

− もしMATCH事象がマッチメーカから受け取られるとすれば(以下のように)オペレーションの処理を開始する。

(c) オペレーションの処理を開始

− APmmconnect()を呼び出し他のオブジェクトマネージャへ接続する。一旦双方のオブジェクトマネージャがAPmmconnect()を呼び出せば、オペレーションへのマッチメーカの介入が終る。

表示し、通常の処理に戻る。

− データを選択するに必要なユーザとのどんな対話でも行う。しかしながら、もしオペレーションがSHAREであり、APmmreserve()を呼び出して他の側が通知されリンクデータを必要としないことを示せば(即ち、SHAREは全体としてオブジェクトへのリンクを作ることであって、オブジェクトのいずれかのデータへのリンクを作成するのではない)、ユーザはデータ選定を急がされるべきでない。

− 情報損失の増える順序で、供給しうる交換フォーマットのリストでAPmmpost()を呼び出す。

− もしAPmmpost()がマッチは不可能であることを示すとすれば(例えば、共通のフォーマットが無く、宛て先はリンクをサポートせず密閉化をさらに不可能とする)、適当なエラーメッセージを表示し、APmmclear()を呼び出し、通常の処理に戻る。

− もしAPmmpost()がマッチの行われることを示すとすれば(即ち、PLACEがすでに通知されて

− 他のオブジェクトマネージャからのリクエストを待機するそのリクエストはリンク、特別の交換フォーマットでのデータあるいは密閉化用かもしれない。

− リンクに対してはリンク仕様を構成し、次いでAPPACKリンク交換ルートを用いてリンク仕様を送る。

− データに対しては、適当なAPPACKあるいは他のデータ交換ルーチンを呼び出し(リクエストされた交換フォーマットに応じて)データを送る

− 密閉化に対しては、

* APrqcopy()を呼び出してオブジェクト全体をコピイするかAPrqcreate()を呼び出して(プロトタイプのコピイを作成し)、かつ(オペレーションにより指示されて)選択したデータを新しく作成されたオブジェクトへ移動あるいはコピイさせる。(オブジェクトのタイプによって、たとえ一部のみをリンクすべきとしても、コピイに全体オブジェクトを含めることが必要かもしれな

いことに注目のこと)。

＊ リンク仕様を新しいオブジェクトの選択したデータに構成し(新しいオブジェクトにおける全てのデータをリンクするとしても必要)、次いでAPPACKリンク交換ルーチンを用いてリンク仕様を送る。

(d) オペレーションを停止する

− もしオペレーションがcopyまたはSHAREであれば、APopfinish()を呼び出しオペレーションを停止する。(SHAREまたはcopyに対するUNDO(取り消し)オペレーションはコンシューマの介入を必要とせず、したがって通信はもはや必要でなくなる)。

− もしオペレーションがMOVEの場合、オブジェクトマネージャは下記の発生までオペレーションを継続させておくこと。

＊ ユーザがMOVEの可能なUNDOを上廻るオペレーションを実行する。この場合、オブジェクトマネージャはAPopfinish()を呼び出しオペレーションを停止し、コンシューマに対して取り消し

− もしAPmapost()がマッチが不可能なこと(例えば共通のフォーマットが無い)を示せば、適当なエラーメッセージを表示し、APmaclear()を呼び出し、通常の処理へ戻る。

− もしAPmapost()が、マッチのあること(即ち、COPY、MOVEまたはSHAREオペレーションがすでに通知されている)を示せば、("PLACE WHAT?"処理を飛ばして)オペレーションの処理を開始する。

− もしAPmapost()がマッチが未だ無いことを示せば、"PLACE WHAT?"の処理を行う。

(b) データ選定を待機("PLACE WHAT?")

− "PLACE WHAT?"を直ちに表示する。

− もしCOPY、MOVE、またはSHAREオペレーションが(このオブジェクトマネージャにより)受け取られたならば、その場合オペレーションは内部的(即ち単一のオブジェクト内での一方の場所から他の場所まで)である。APmaclear()を呼び出しマッチメーカをクリヤし、オペレーションを進める。

(43) はもはや不可能となることを通知する。

＊ ユーザがオペレーションのUNDOを要求する。この場合、オブジェクトマネージャはコンシューマと通信して運動したデータを検索し、MOVEオペレーションを反転する。

マッチの起る前にCANCEL事象が受けとられた場合、APmaclear()を呼び出しマッチメーカをクリヤし、通常の処理に戻る。

15.2.2. 場所プロトコル

オブジェクトマネージャがPLACE事象を受け取ると、オブジェクトマネージャは次のステップを踏む。

(a) オペレーションを通知する。

− APmareserve()を呼び出して同じタイプのオペレーションがすでに進行中であるか検出する。そうであれば、エラーメッセージを表示し、通常の処理に戻る。

− APmapost()を、情報損失の大きい順の、受け入れられる交換フォーマットのリストで呼び出す。

− もしMATCH事象がマッチメーカから受け取られると、(以下のように)オペレーションの処理を開始する。

(c) オペレーション処理の開始

− APmaconnect()を呼び出し他のオブジェクトマネージャへの接続を行う。一旦双方のオブジェクトマネージャがAPmaconnect()を呼び出すと、そのオペレーションへのマッチメーカの介入は終了する。

− 原始オブジェクトマネージャに、リンク、特別の交換フォーマットでのデータあるいは密閉化のリクエストを出す。

− リクエストが確認された後、APPACKの1つあるいは他のタイプのデータ交換専用サービスを用いて交換を遂行する。

− もしリクエストがマルチ(多重)オブジェクト転送(APDATAMULTIPLE)のためのものであるとすれば、以下の余分のステップが必要とされる。

＊ APmareserve()を呼び出しマルチオブジェクト転送用にマッチメーカを予 する。

＊ APPACKのマルチ交換サービスを用いて、オブジェクトの供給されたリストを通してループ化し、各々とインタラクトしてその中味を探索する。

(d) オペレーションを終了する

－ オペレーションがCOPYまたはSHAREであれば、APopfinish()を呼び出してオペレーションを終了させる。

－ もしオペレーションがMOVEであれば、オブジェクトマネージャは下記の発生するまでオペレーションを継続させるべきである。

＊ ユーザが、MOVEの可能性のあるUNDOを上廻るオペレーションを実行する。この場合、オブジェクトマネージャはAPopfinish()を呼び出してオペレーションを終了させ、コンシューマに取り消しがもはや可能でない旨通知する。

＊ ユーザがオペレーションのUNDOをリクエストする。この場合、オブジェクトマネージャはコンシューマと通信して移動したデータを検索してMOVEオペレーションを反転させる。

－ もしオペレーション終了メッセージが受け取られるとすれば、次にAPopfinish()を呼び出す。MOVEの一方の側のみが取り消ししうる。もしMOVEが抑えられる前にUNDO事象が後で受け取られるとすれば、ユーザは、MOVEの半分のみが取り消されうることを警告され、確認を促がされる。

16. データ交換(第９図)

データ交換に関する以下の説明において説明するように、オブジェクト間のデータの交換は原始オブジェクトおよび宛て先オブジェクトにおけるデータのタイプによって左右される。もしデータタイプが同一か、十分近似しているとすれば、データをほとんどあるいは全く交換することなく、かつ情報をほとんどあるいは全く損失することなく直接交換され、かつ宛て先において直接用いることができる。しかしながら、もしデータタイプが十分相違しているとすれば、変換は困難であり、また、データを宛て先オブジェクトで直接使用できない、即ち宛て先オブジェクトのデータの一体部分として使用できなくなる。この場合、情報に

15.2.3. オブジェクト間のMOVEオペレーションの後のUNDOの処理

オブジェクト間のMOVEオペレーションを取り消すために、原始および宛て先の双方のオブジェクトマネージャはUNDOがリクエストされるか、オペレーションが(例えば別のオペレーションにより抑えられて)もはや反転できなくなるまでオペレーションを継続させておく必要がある。オブジェクト間の(外部の)MOVEが依然として作用している間オブジェクトマネージャは以下のことを行う。

－ もしUNDO事象がユーザから受け取られると、他のオブジェクトマネージャにMOVEUNDOリクエストを送りMOVEの反転に進行する。

－ もしMOVEUNDOメッセージが他のオブジェクトマネージャから受け取られると、MOVEを反転するよう進行する。

－ もしユーザが外部のMOVEを、現在反転可能のオペレーションとして上廻るオペレーションを実行するとすれば、APopfinish()を呼び出してオペレーションを終了する。

若干の損失があり、後述するように原始オブジェクトのデータは出来るだけ上手く宛て先オブジェクトのデータに「はめ込む」必要がある。

オブジェクト間のデータ交換は「マッチメーカ」と称されるAPPACK装置により促進することができる。マッチメーカは２個のオブジェクトマネージャが、それらがデータ交換オペレーションの終りをマッチングさせていることを検出できるようにする。原始および宛て先オブジェクトマネージャとマッチメーカとのインタラクション(対話)の一部として、これらのオブジェクトマネージャはマッチメーカに対して、オブジェクトがデータを提供し、かつ受け入れうるフォーマット即ち形式を示す。これらのフォーマットと形式とはオブジェクトの元のデータタイプを含みうる。各オブジェクトに対するオブジェクトマネージャはまた、ある種のデータ変換を行う能力を有し、したがって、それらの元のフォーマット以外のフォーマットや形式でデータの提供あるいは受け取りができる。交換用データフォーマットは原始および宛て先オ

ブジェクトマネージャとマッチメーカとの間の対話により合意され、交換は、どのようなデータ交換がいずれかのオブジェクトで必要とされようとも合意したフォーマットにより実行される。

リンクを介するデータ交換において、子供のオブジェクトのデータを提供すべきデータタイプは、特定のリンクに対する後でのデータ交換において子供オブジェクト404が後で使用するようリンクのリンクパラレルファイルエントリ404の交換タイプフィールドにおいて記録される。マッチメーカは最初にリンクを作成する共用オペレーションには介入しうるものの、リンク更新オペレーションでは介入しない。

第９図を参照すれば、データが原始オブジェクトから宛て先オブジェクトまで移動され、コピイされ、あるいは共用される、APPACK218のデータ交換が線図表示されている。原始ユーザオブジェクト232-sと関連のオブジェクトマネージャプログラム240-s、宛て先ユーザオブジェクト232-dとその関連のオブジェクト／マネージャプログラム

804-d、フォーマットテーブル806-dおよびデータ転送／変換ルーチン808-dを含む。

さて、データ交換オペレーションを検討すると、前述の説明ではデータ交換はデータのコピイおよび移動オペレーション並びにデータ共用に使用できるということであった。前記オペレーションの全ては多くの点において類似していることにまず注目すべきである。コピイ、移動および共用オペレーションに関する以下の説明でさらに詳しく説明するが、これらオペレーション間の主たる差は、それらオペレーションが開始される態様と、データがオペレーション後扱われる態様とにある。

後者に関しては、３種のオペレーション、即ちコピイ、移動および共用オペレーションの全ては基本的にはコピイオペレーションと見做すことができる。即ち、各オペレーションにおいて、あるデータが識別され、原始オブジェクトからコピイされ、宛て先オブジェクトの識別した位置に置かれる。移動オペレーションとコピイオペレーションの相違点は、原始オブジェクトにおけるデータ

240-d、およびアプリケーションパック(APPACK)218とが示されている。ユーザオブジェクト232-sは原始データ800を含むものとして示され、該データはコピイされ、移動され、あるいはユーザオブジェクト232-dと共用されるデータであり、一方ユーザオブジェクト232-dは宛て先データ802を含むものとして示され、該データは交換されたデータがユーザオブジェクト232-dで占める場所と交換されたデータ自体の双方を示す。

オブジェクト／マネージャプログラム240-sおよび240-dを参照すれば、これらオブジェクト／マネージャプログラム240の各々はそれらの各々のユーザオブジェクト232にオペレートしているものとして示されている。各オブジェクト／マネージャプログラム240-sの実行可能コードはデータ交換ルーチン804-sを含むモジュール、関連のフォーマットテーブル806-sおよびデータ転送／変換ルーチン808-sを含むモジュールを含むものとして示されている。同様に、オブジェクト／マネージャプログラム240-dはデータ交換ルーチン

がオペレーションの終りにおいて削除される、即ちデータが原始オブジェクトから宛て先オブジェクトへコピイされ、原始オブジェクトにおけるデータの原コピイが削除されることである。共用オペレーションも、原始オブジェクトからのデータの新しいコピイが作られ、前述のようにリンクが更新される毎に宛て先オブジェクトに位置される点を除いてコピイオペレーションと類似である。

データ交換オペレーションの開始に関して、最初のステップはオペレーションの開始と、データが宛て先オブジェクトにおいて現われる位置の識別と、および原始オブジェクトにおいて交換されるべきデータの識別とである。後述のように、コピイ、移動および共用オペレーションは、通常メニューピックアップあるいはキーボードを介して入力される指令によりシステムのユーザが典型的にはマニュアルで開始させる。次に、原始オブジェクトデータおよびデータの宛て先オブジェクトにおける位置との識別の順序は、ユーザの見方で原始オブジェクトと宛て先オブジェクトとの間

を移動しているオペレーションがオペレーション「から」来　のかオペレーション「へ」向うのかによって変わり、オペレーションの順序(即ち、ユーザがまず原始データを識別できるのか、あるいはユーザがまず宛て先データでの位置を識別できるのか)によって変わる。

さらに、リンクを介するデータ交換もリンク更新の発生時に、即ちマニュアルの更新が選択されていた場合はユーザの指令により、あるいは自動的更新が(共用オペレーションが最初にリンクを作成すると発生する更新に加えて)選択されていたとすればリンクされたデータを含む宛て先オブジェクトの部分のディスプレイあるいはプリントによって開始される。

原始オブジェクトのデータのユーザによる選択と、データが行くべき宛て先オブジェクトでの位置の識別とに関して、原始および宛て先オブジェクトマネージャが呼び出される。

次いで、本明細書の説明の主眼であるが、原始および宛て先オブジェクトは原始オブジェクトか

ジェクトのオブジェクト／マネージャプログラム240のデータ転送／変換ルーチン808により発生され、かつ受け入れられうる全てのデータフォーマットのリスト即ちテーブルを含む。これらのフォーマットは少なくともオブジェクトの元のデータフォーマットを含み、かつある外部のデータフォーマットを含みうる。

各オブジェクトマネージャのデータ交換ルーチン804はマッチメーカ810に対して、データ交換ルーチン804の関連のデータ転送／変換ルーチン808が受け入れ、あるいは提供しうる各種のデータフォーマットを記述した、関連のフォーマットテーブル806からの情報を提供する。これらのフォーマットは好ましい順に提供され、マッチメーカ810が交換のための共通のデータフォーマットを選択する。マッチメーカ810は交換のために選定したデータフォーマットをデータ交換ルーチン804へ戻し、次いでデータの転送が行われる。

データ交換における次のステップは原始オブジェクトから宛て先オブジェクトへデータを実際に転

(46) ら宛て先オブジェクトへデータを交換する必要がある。第９図に示す本説明では、ユーザの行動(即ち宛て先の位置の識別)がすでに実行されているものと想定する。

データ交換オペレーションの開始時、原始および宛て先オブジェクトのオブジェクトマネージャは、データ交換オペレーションを導き、かつ制御するルーチンのグループからなる各データ交換ルーチン804を呼び出す。データ転送／変換ルーチンモジュール808とそれに関連したフォーマットテーブルとがある。データ転送／変換ルーチン808はオブジェクトとAPPACK218のデータ交換装置812との間でデータを転送する、即ちオブジェクトから、かつオブジェクトへデータを読取ったり、書込んだりするための、一群のルーチンから構成されている。また、データ交換装置812は、オブジェクトの元のデータフォーマットとある外部のデータフォーマットとの間でデータを変換するデータ変換ルーチンも含む。フォーマットテーブル806の方は、対応するオブジェクトにより、即ちオブ

送することである。転送はストリームモードで実行される。即ちバイトが転写オペレーションにより相互に区別されず、データのバイトの基本的に制限のないストリームの形で転写される。即ち、ストリームのバイトに対する重要性が、交換オペレーション自体でなく、交換フォーマットにより規定される。

データストリームの転送を実施するために、各々のデータ転送／変換ルーチン808は、それぞれのユーザオブジェクト232とAPPACK218のデータ交換装置812との間で一連の「GET」および「PUT」オペレーションを実行する。即ちデータ転送／変換ルーチン808aは原始データから一連のバイトのデータを「GET(取得)」し即ち読取り、原始データが原始データ800から「取得」されるにつれて必要なデータ変換オペレーションを実行しながら、集められた原始データのバイトをデータ交換装置812に「PUT(置く)」即ち書込む。データ転送／変換ルーチン808-dは同時に、データ交換装置812から原始データのバイトを「取得」即ち読取り、再びデータ

が宛て先データ802へ「置かれる」につれて必要なデータ変換オペレーションを実施して原始データのバイトを宛て先データ802へ「置く」即ち書込む。この点に関して、データ交換装置812は一組のレジスタと、例えばデータ転送／変換ルーチン808-aのような原始データを受け取り、データ転送／変換ルーチン808-dのようにデータを宛て先へ転送するデータチャンネルを含むルーチンとから構成されていることに注目すべきである。

さらに、データ転送／変換ルーチン808のオペレーションが例えばリンクマーカのような各種タイプのはめ込みデータを含めることをサポートすることに注目すべきである。ストリーム転送に採用されている規則は、ストリームに対して初期の変換タイプ即ちデータフォーマットがあることである。例えばリンクマーカあるいはリンクされたデータのような別のデータタイプをストリームに挿入すべき場合、データタイプの変更と新しいフォーマットとを示すインジケータコードがストリームに挿入され、適当なデータ転送／変換ルー

(47) チン808のデータフォーマット変換装置が呼び出されてデータ転送／変換ルーチン808のオペレーションを新しいデータタイプに切り換える。データタイプが例えばデータストリームの挿入された部分の終りにおいて再び変更されると同じオペレーションが用いられ、データ転送／変換ルーチン808は元のデータタイプに戻る。

最後に、前述の説明から、データ変換フォーマットは、リンク用であろうと移動あるいはコピイオペレーション用であろうと、あるいは考えうるいずれからの他のタイプのデータ交換に対してであっても、データ交換のあらゆるタイプに対しても中心的であることが明らかである。前述のように、一旦２つのオブジェクトマネージャが共通のデータ交換フォーマットについて合意すると、原始オブジェクトマネージャは原始オブジェクトの内部データフォーマットからの原始データを必要に応じ交換フォーマットに変換し、データを交換フォーマットにおける宛て先オブジェクトマネージャへ伝送する。宛て先オブジェクトマネージャ

はデータを受け取り、必要に応じて変換フォーマットからのデータを宛て先オブジェクトの内部データフォーマットに変換し、即ちデータを宛て先オブジェクトに内部化し、データを宛て先オブジェクトに置く。さらに、交換に含まれるフォーマットの複雑さ、「リッチさ」および互換性が交換の複雑さあるいは容易さ、およびあるとした場合の情報損失あるいは歪の可能性を規定する。

各オブジェクトマネージャはそれが内部的にオペレートするデータのフォーマットや構造を決めることができる。このため各オブジェクトマネージャの最も強力で効率的な実行性を許容する。オブジェクトマネージャは典型的に内部使用のためのシステム定義のデータフォーマットを使用しない。

いずれか所定の対の内部データフォーマットは異なる属性を有しており、所定の原始データフォーマットと所定の宛て先データフォーマットとの交換フォーマットを選択することにより、情報損失あるいは歪みの可能性も含みデータ交換の速度、

効率および正確さに直接影響を与えることが上記の説明から明らかである。データの中味の保持、即ち所定のデータフォーマットの属性を表現あるいは変換する能力はデータ交換を達成する上でシステムのユーザに対して最も重要な要素の１つであり、したがって、データ中味を可能な限り最も高度に保持する交換フォーマット、即ち原始および宛て先の内部データフォーマットの属性を最大限収容しうる交換フォーマットを用いて各データ交換を実行することが重要である。そのため、各オブジェクトマネージャは出来るだけ高レベルの交換フォーマットをサポートし、かつ出来るだけ多くの交換をサポートすべきである。このため、所定の各データ交換に対して最も効率的な交換フォーマットを利用できる確率を増し、かつ所定のオブジェクトマネージャが高レベルでデータ交換しうるオブジェクトマネージャの数と範囲とを増大する。このことによりまた、オブジェクトマネージャ間およびオブジェクトマネージャとシステムとの間の結合の程度を向上させる。

交換フォーマットは3レベルに構成される。交換フォーマットレベルの区別は交換において達成されるデータ中味の保持程度、即ち交換が原始および宛て先の内部データフォーマットの異なる属性のサポートしうる度合いに基いている。3種類のフォーマットレベルはそれぞれデータおよび属性の保持レベルの大きい順に「バニラ」フォーマット、「カテゴリ特定」フォーマット、および「専用」フォーマットと指示されている。

まず「バニラ」データ交換フォーマットを検討する。これらフォーマットは、システムに現われうるデータタイプの全スペクトルの間で、但しデータ交換が過度に複雑にならぬレベルでデータ交換を提供する意図のものである。基本的に「バニラ」フォーマットは、例えばASCⅡあるいは2進ファイルのように、システムに現われうる全てのデータタイプの中の最低の共通の特徴を示す一般的なデータ交換フォーマットである。即ち、一般的なフォーマットは種々の内部データフォーマットの実用的な最大数の内部データ属性の最大の共通

タイプの固有の制限内での最大限のデータの中味や属性の範囲をサポートするように構成されていることに注目すべきである。この点に関して、一般フォーマットが宛て先オブジェクトが受け入れ可能な以上の情報を含みうることもある。しかしながら、一般フォーマットは、宛て先のオブジェクトマネージャがそれが使用しえない、あるいは解釈してないデータ中味の部分は無視して各宛て先オブジェクトが提供されたデータを最大限利用できるように構成されている。

現在定義されているバニラ交換フォーマットの例としては、定義されたレコードフォーマット、未定義のレコードフォーマット、テキストフォーマット、ビクチャフォーマット、リンクフォーマットおよびサウンドフォーマットを含む。これら一般フォーマットの各々を検討すれば、定義されたレコードフォーマットは主として数字あるいはテキストである各種タイプの情報を含みうるフィールドを含む、レコード構造の定義を可能とする。このフォーマットは例えば詳細なデータベース変

(48) セットをサポートする。これらの一般的な交換フォーマットは、2個のオブジェクトマネージャに関連のデータが基本的に異なるタイプである場合、即ち内部データフォーマットがそれらの属性において著しい程度の共通性を有さない場合使用する意度のものである。その例としては、グラフィックタイプのオブジェクトとドキュメントタイプオブジェクトとの間の交換あるいはスプレッドシートあるいはデータベースタイプオブジェクトとドキュメントタイプオブジェクトの間、あるいはスプレッドシートあるいはデータベースタイプオブジェクトとグラフィックタイプオブジェクトとの間の交換である。

これらの場合において、データの内部表示およびデータのユーザモデルの内部表示は、宛て先のオブジェクトが原始オブジェクトデータ表示の、即ち原始オブジェクトデータの相対的に小さい細部分以上のものは解釈あるいは表示しえないため高レベルの交換をしようとも単純には実用に供さない。しかしながら、一般フォーマットはデータ

交換あるいはスプレッドシートフォーミュラ交換を意図したものでなく、レコード向きのオブジェクトマネージャの間で有意な情報の転送ができるようにしたものである。

未定義のレコードフォーマットは前述の定義したレコードフォーマットの変形であるが、レコード記述子は含まず、そのため各レコードの中味はレコードによって変りうる。この未定義のレコードフォーマットは、あるオブジェクトマネージャが未定義のフォーマットのデータを拒否したいと思うこともあるため定義されたレコードフォーマットとは明確に区別されている。

テキストフォーマットは例えばASCIIストリングフォーマットにおけるようにテキストデータを単純に表示するが、あるテキストの属性情報と単純なフォーマット化情報は含むことができるものである。さらに、テキストフォーマットは例えば定義されたレコードフォーマットあるいは未定義のレコードフォーマットにフィールドの中味として含入できる。

ピクチャフォーマットはオブジェクトマネージャが、グラフィック能力がほとんど無い、あるいは皆無の宛て先オブジェクトにより表示しうる表示可能の「エンティティ」を提供できるようにする意図のものである。テキストオブジェクトにリンクされたグラフィックデータの場合、例えば、グラフィック「エンティティ」はリンクパラレルファイルエントリ408のリンクされたデータコピイ458に記憶され、宛て先テキストオブジェクトにおいてアイコン、あるいはタイトルで表示されることによりテキストオブジェクト自体ではグラフィックデータとして実際には介在しない。次いでグラフィックデータは、グラフィックデータを「含む」テキストオブジェクトの部分を表示すべき場合、表示のために、リンクされたデータコピイ458から読みとられる。このタイプのフォーマットの意図は合理的に完全な可視性中味を有するが、意味あるいはグラフィックデータを編集する能力の無いグラフィック表示性を提供することである。

リンクフォーマットは、リンクのコピイと移動

(49) オペレーションについて前述のように、オブジェクトマネージャがリンクを交換できるようにさせるものである。リンクフォーマットは基本的には、前述のリンク構造を表示する。

サウンド(音響)フォーマットはピクチャフォーマットと類似であるが、グラフィックデータではなく、例えば音声メッセージのようにサウンドデータを交換するためのものである。このフォーマットも、合理的に完全なオーディオの中味を以ってサウンドデータを交換したり、「再生」できるが、意味的な中味あるいはサウンドデータを編集する能力は無いものである。

カテゴリ特定フォーマットを検討する。「カテゴリ特定」交換フォーマットは類似のデータモデルを有するが内部データ記憶フォーマットの異なるオブジェクトの間でデータの属性の著しく大きい中味と著しく大きい保持力としてデータ交換する意図のものである。例えば、種々のスプレッドシートオブジェクトは類似のデータモデルを有するが、内部データ表示は極めて相違する。そのた

め、２個の異なったスプレッドシートは一般フォーマットより著しく大きい中味でデータ交換できるが、それらの内部データフォーマットでは直接データ交換できない。

各カテゴリ特定交換フォーマットは、そのカテゴリ特定交換フォーマットを用いるカテゴリの全てのオブジェクトマネージャに有意なデータと記述情報とを含む。この点に関して、カテゴリの定義とは、カテゴリは所定のカテゴリ特定フォーマットにおいてデータを提供、受け取り、あるいは提供と受け取りの双方を行うことを合意した１つ以上のオブジェクトマネージャからなるグループである。

カテゴリ特定交換フォーマットの可能な数は、基本的には無制限であって、新しいカテゴリ特定交換フォーマットを備えた新しいカテゴリは意のままに作成できる。例えば、もし一般フォーマットより大きい中味のデータを交換することを２個の異なるオブジェクトマネージャが必要あるいは希望することが判明した場合、それらのオブジェ

クトマネージャに対してカテゴリ特定フォーマットを作成し、新しいカテゴリを定義することができる。次いでいずれの数の、他の異なるオブジェクトマネージャも新しいカテゴリ交換フォーマットを使用するよう選択し、そのカテゴリのメンバとなればよい。この点に関して、いずれの所定のオブジェクトマネージャもいずれかの数のカテゴリのメンバとなりうることに注目すべきである。

定義されたカテゴリ特定フォーマットの例としては、ウォング情報転送アーキテクチヤ(Wang Information Transfer Architecture－ＷＩＴＡ)、データベースフォーマット、スプレッドシートフォーマットおよびグラフイックフォーマットを含む。再びこれらフォーマットの各々を検討する。ＷＩＴＡフォーマットはマサチューセッツ州ロウエルのウォング・ラボラトリーズ社(Wang Laboratories, Inc. of Lowell, Massachusetts)により開発されたＷＩＴＡデータ交換アーキテクチヤをさらに開発したものである。本発明のシステムにおいては、ＷＩＴＡフォ

ーマットは、広範なフォーマット化情報および可能性として各種タイプのその他のオブジェクトへのリンクを含むテキストデータを主として交換する上でドキュメントやテキストエディタが使用する意図のものである。ＷＩＴＡはまた、一方のドキュメントタイプから他方のドキュメントタイプへテキストおよびフォーマット化情報を交換するのに特に適しており、一般的なデータ通信に使用される。

データベース交換フォーマットはデータベースタイプオブジェクトマネージャが使用する意図のものである。このフォーマットは、例えばデータベース定義、レコード構造、レコード間の接続についての情報を有するデータレコード、計算されたフィールドとフォーミュラフィールドおよびリンクを含む考えうる全てのデータベースタイプのデータを交換する上で使用する意図である。データベースフォーマットはまた、例えばドキュメントとスプレッドシートオブジェクトのあるタイプのものを交換したり、通信するのにも使用しうる。

スプレッドシート交換フォーマットはスプレッドシートタイプのオブジェクトマネージャが使用する意図のものである。スプレッドシート交換フォーマットにおいて交換されたり変換されるスプレッドシートデータは、例えば行、列、およびセルの定義、公式およびテキストおよびデータフィールドを含む全てのタイプのスプレッドシートタイプデータを含む。

最後に、グラフイック交換フォーマットはグラフイックタイプのオブジェクトマネージャが使用する意図のものである。このフォーマットは、例えばビットマップ化したグラフイックデータ、ベクトルおよび「オブジェクトベースの」グラフイックおよびピクチャ情報の全ての形態を含む考えられる全てのグラフイックタイプのデータを交換する上で使用する意図のものである。（「オブジェクト」という用語はここでは本明細書の残りの部分で使用する意味でなく、グラフイックに関係した視点から使用していることに注目すべきである。）

最後に「専用」交換フォーマットは正に同じデー

タ構造とフォーマットとを有するデータオブジェクト間でのデータ交換に用いられる。これらの場合、交換に参加するオブジェクトマネージャの各々は、正確に何が他のオブジェクトマネージャの内部データ表示や属性であるか知っており、したがって最大のデータ中味を保持するためにデータをどのようにパッケージにし、かつ通信すべきか知っている。データ構造は基本的に同じであるべきとの要件のために、専用交換フォーマットは同じオブジェクトマネージャにより作成されたデータオブジェクト間のデータ交換に最も一般的に使用される。

専用交換フォーマットにおいては、オブジェクトマネージャは、いずれかの他のオブジェクトマネージャの内部データ構造との互換性の必要が全くないため自由にそれ自体の専用の交換フォーマットを定義すればよい。しかしながら、専用の交換フォーマットを有しているオブジェクトマネージャはその専用フォーマットを発表して、他のオブジェクトマネージャかその専用フォーマット

を使用するよう選択することによって、新しいカテゴリ特定の交換フォーマットを作成できる。専用フォーマットはオブジェクトマネージャの内部データ構造により細部に定義されるので、専用フォーマットをカテゴリ特定フォーマットに変えても、オブジェクトマネージャが後でその内部データ構造を変えうる範囲には制限がある。

所定のオブジェクトマネージャはデータオブジェクト間で通信するために専用の交換フォーマットを使用する必要はないが、カテゴリ特定のフォーマットを使用するよう選択でき、あるいは一般フォーマットでの交換に十分なこともありうることに注目すべきである。さらに、所定のいずれかのオブジェクトマネージャは同時に、専用変更フォーマットを有し、いずれかの数のカテゴリに属し、かつ対応するカテゴリ特定フォーマットを用いたり、かつ一般フォーマットを用いてもよい。

最後に、一般、カテゴリ特定あるいは専用フォーマットであろうとも、各交換フォーマットに対して、どのようにデータを並べるか、かつどのよ

うにデータ交換を行うかオプションによって決めるいずれかの数の交換オプションのあることに注目すべきである。宛て先オブジェクトマネージャの決めるオプションや原始オブジェクトマネージャが決めるオプションがある。所定のデータ交換に対して、いずれのオプションを選択するかは交換の行われるときに原始および宛て先オブジェクトマネージャによって決められ、特定の交換フォーマットを使用する全てのオブジェクトマネージャは出来るだけそのオプションを使用するものと思われる。

17. 資源

前述のように、オブジェクトマネージャは、実行可能のコードと、プログラムにより使用されるデータとから構成されているものと見做しうる。そのようなデータの例としてはアイコン、言語依存テキスト、メッセージ、テキストフォント、フォーム、日付および時間の表示フォーマットを含みうる。本発明によるシステムは、プログラムのコードとは別に変更あるいは共用したいと希望

(51) するデータを記憶するために使用しうる「資源」を提供する。したがって、資源はそのようなプログラムを、それが関連しているプログラムとは別に、かつ独立して記憶する手段である。

資源にデータを位置させることによりその中の情報をプログラムの実行可能コードに影響を与えることなくカストマイズあるいは変更できるようにする。この手段により、所定のプログラムに対して多くのバージョンができ、実行可能コードは各バージョンに対して同じであるが関連の資源の常駐する該プログラムのデータのみがバージョンによって変わる。さらに、資源を用いることによりオブジェクトマネージャと他のプログラムとが共通のプログラムデータを共用することにより個々のプログラムのサイズとメモリ要件とを減少させることができる。

17.1. 資源ファイル

資源は典型的には多く資源ファイルに記憶される。資源は所期の関連に基き資源ファイルにグループ化される。資源を用いたプログラムは、一時

に開放されている複数の資源ファイルを有する。資源がリクエストされるとそのプログラムの現在開放している資源ファイルが、リクエストされた資源ＩＤを有する資源に対して検索される。このようにどの資源ファイルをプログラムが開放するか否かにより種々の資源を使用できる。

以下の例はファイルに資源をグループ化することに係る事象のあるものを示す。１個の資源ファイルは、英語でスプレッドシートオペレーションに特に向けられるユーザのプロンプトの全てを含みうる。第２の資源ファイルは仏語におけるプロンプトを含む、第３と第４の資源ファイルはそれぞれ英語と仏語の一般的なプロンプトを含む。スプレッドシートオブジェクトマネージャは典型的にこれら資源ファイルの２個を開放している。第１と第３のファイルは英語を話すユーザ用であり、第２と第４のファイルは仏語を話すユーザ用である。また、ドキュメントオブジェクトマネージャはドキュメントオペレーションに特有の資源を含む資源ファイルと共に第３または第４の資源ファ

イルのいずれかを開放する。

資源ＩＤは資源ファイル内で一意的であるが、種々の資源ファイルを介して一意的である必要はない。資源ＩＤのある範囲が一定の定義を有するように割り当てられる。換言すれば、同じＩＤを有する１個以上の資源があるかもしれないが(例えば一方が英語資源で他方が仏語資源のファイルにあるが)、これらの定義が一定の資源ＩＤsの中の１個を共用する全ての資源は同じ目的である(例えば同じプロンプト)。完全に異なる目的に対して他の資源ＩＤsを異なるプログラムにより使用しうる。スプレッドシートに関連の資源のファイルはデータベースに関連の資源のファイルに使用されるように同じ資源ＩＤを使用しうる。

各資源ファイルは３種類のセクション、索引付情報、資源データ、記述情報から構成される。

索引付情報は資源ファイルの前方に記憶される。これは、希望する資源がファイルにおいて探されたか否かを検出するに必要な情報を含む。この索引は、一連のアレイとして構成され、各アレイは

順次の資源ＩＤのクラスタに対応するエントリを含む。資源ファイルの索引部分の各エントリは

－資源のサイズ、

－資源ファイル内の資源の位置、

－資源タイプコード、および

－対応する資源がカストマイズできるか否かを指示するフラッグを含むフラッグを含む。

資源ファイルの資源データ部分はいずれかの順序で資源を記憶しうる(資源の各々の位置はファイルの前出し部分に指示される。共に使用されそうな資源は相互に対して物理的に近接させて記憶することが好ましい。このため、リクエストされる資源がすでに主メモリにあり、ディスクでのアクセスを低減させる可能性を増大できる。

記述情報はオプションのものである。これは資源のテキスト名、資源のテキスト記述およびリテラルとを含みうる。

(任意的に介在させうる)リテラルは資源ＩＤsを備えた資源を探すプログラムの原始コードでシンボルを突き合わせるために使用される。このため(52)プログラム原始コードが、その簡略値について選択し、かつ資源ＩＤの簡略値と独立したシンボルを備えた資源を参照するようにできる。ユーティリティプログラムが資源ファイルを読み取り、かつプログラムヘッダファイルを作成する。資源ファイルにおいてリテラルが定義される各資源に対して、エントリが、その値が対応する資源ＩＤであるシンボルにリテラルがなるよう定義するヘッダファイルに現われる。このように、これらの機械的に発生したヘッダファイルの１個を含むプログラムが、資源を識別するために資源ＩＤでなくむしろリテラルを使用することができる。

17.2. 資源(第10Ａ図)

第10Ａ図を参照すれば、本システムの全ての資源を代表する資源700の概図が示されている。全ての資源700は、RESPACK222のルーチンを介して資源700をシステムが位置づけ、かつアクセスする情報を含むヘッダ702と、実際の資源データを含む資源の中味704とからなる共通の構造を共用する。以下説明するように、その中に含まれる情

報は資源を作成したエディタにより検出されるものの、資源の中味704の内部構造が資源のタイプにより検出される、資源のある定義されたタイプがある。また、資源の中味704の中味と内部構造とは資源を作成したエディタにより全体的に検出される自由形態の資源のタイプもある。

各種タイプの資源の資源の中味704の構造と、中味とは資源700のタイプによって変わるが、全ての資源700のヘッダ702は構造ならびに中味が均一である。指示しているように、ヘッダ702はタイプフィールド708、名前フィールド708、識別子フィールド710、リテラルフィールド712、記述フィールド714、一致したフラッグフィールド716としてそれぞれ指示される３個のフラッグフィールド、ユーザがカストマイズしうるフラッグフィールド718およびシステムが修正可能のフラッグフィールド720とを含む。

タイプフィールド706は、例えば資源がフォーム、メッセージ、プルダウンメニュー、アイコン、ウインドウ構造、あるいは例えばテキストあるいはデータのバイトストリングであろうと、特定の資源700に含まれた資源のタイプの識別子を含む。この点に関し、フォームとは典型的にはスクリーンの表示、即ち、有益で明確なテキストフィールドおよび多くの場合ユーザにより埋めることの可能なフィールドを含むＣＲＴディスプレイである。メッセージの方は、例えばイベントあるいはユーザが実施すべき行動の通知のような情報をユーザに提供する手段であって、グラフイック、テキストあるいはオーディオでよい。周知のようにメニューとは、有限の複数の可能な行動あるいは事象の間でユーザが選択できるようにする手段である。プルダウンメニューは、そのメニューが必要なときのみ、即ちユーザの選択あるいはメニュー表示が適当であるとオブジェクトマネージャが検出すると実行されるものである。「プルダウン」とは単に、ニューが明らかにスクリーンの上部から引き下げることを意味する。アイコンはテキストストリングの代りにグラフ表示するものであって、テキストがメッセージを提供したり、行動あるい

は選択を表示し、あるいは何かを識別するために使用されるほとんどの何処の場所においても使用しうる。ウインドウ資源は、ウインドウ、即ちユーザへの可視表示を含むスクリーンの有限の部分を説明するパラメータブロックであり、ユーザに表示すべき対応の窓を発生させる「パック」により使用される。

その他の定義された資源タイプはタイプフォントを定義するフォント資源、特定の言語の統語および文法上のルールを定義する言語ルール資源と、キーボードの各種のキーの機能と用途とを定義するキーボード変換テーブルとを含みうる。最後に、バイトストリング資源は前述の定義したタイプの中の１個の範囲内にない何らかの資源を記憶するための自由にフォーマット化されるバイトストリングである。このタイプの資源において、情報の内部構造は資源を作成するエディタにより完全に検出される。

名前フィールド708は資源700に対する任意に識別する名前を割り当てするために使用される。名

(53) 前フィールド708に常駐す　名前は例えば指令言語を介して資源700をアクセスするために使用しうる。

識別子フィールド710は、識別子がプログラムの実行可能コードにより参照されると資源にアクセスするためにRESPACKにより使用される識別子を含む。プログラムが資源を必要とすると、プログラムは資源ＩＤをRESPACKルーチン(例えばRESget())に通す。資源識別子を用いて、RESPACKの方は現在開放している資源ファイルの１個で資源を位置づけ、資源をプログラムに提供する。

識別子フィールド710に常駐する識別子は、資源がグループ化されているファイル内においてのみ一意的であればよい。典型的には識別子は資源が使用されるべきときのみ資源ファイルのセット内で一意となる。

リテラルフィールド712は、資源を用いるプログラムの原始コード内で識別子フィールド710の資源ＩＤを言及するラベルとして使用されるテキストストリングを含む。資源エディタの方は所定

の資源ファイルに対応する「INCLUDE(含入)」ファイルを生成する機能を提供する。

記述子フィールド714は資源の表示可能の記述からなるテキストストリングを含む。記述子テキストストリングは、資源エディタの資源のインデックス(索引)に、かつ資源を用いているユーザあるいはオブジェクトマネージャに対してはユーザあるいはプログラムプロフイルに表示できる。

最後に、ヘッダ702のフラッグフィールドを検討すると、一致したフラッグフィールド716は、セットされると資源が資源ファイル内のブロック境界に整合させるフラッグを含む。本発明のシステムにおいては、ブロック境界に整合しないのは新しく資源を作成した省略値である。しかしながら、整合は例えばフロントのようなある資源に対して有利であって、あるいはもし資源ファイルが大きくて、多くの資源を含んでおれば、あるいは資源自体が大きい場合、効率のためにも望ましい。

ユーザがカストマイズしうるフラッグのフィールド718とシステムが修正可能のフラッグのフィ

ールド720とは、双方共、資源のカストマイズしたコピイが作成しうるか否か、および誰によって作成されうるかを示すフラッグを含んでいるという点において類似である。ユーザがカストマイズしうるフラッグのフィールド718は、ユーザが資源のカストマイズしたコピイを作成しうるか否かを指示す。システムが修正可能のフラッグのフィールド720は同様であるが、システムの管理者がシステムプロファイルにおいて資源をカストマイズしうるか否かを示す。

前述のように、資源の中味704は、資源データの構造が資源のタイプにより決められる実際の資源データを含む。さらに、この点に関して、所定の資源は１個以上の他の資源から構成しうることを注目すべきである。即ち、資源700の資源の中味704は基本、即ち複数の他の資源700の識別子710を含むことによってそれらの他の資源700も組み入れればよい。例えば、フォームタイプの資源は、各グループのフィールドが資源700であるフィールドのグループから構成すればよい。各グループ

のフィールド、即ち、グループのフィールドを定義する資源700の各々は別のフィールドあるいはフィールドのグループから構成してよく、これらのフィールドの方は基準、即ちメッセージタイアの資源を識別する識別子710を含めばよい。

資源には複数のタイアがあり、所定の資源700の一部の資源の中味704の構造と中味とは資源のタイアにより変わる。しかしながら全ての資源700のヘッダ702の部分は構造並びに中味が均一であって、タイアのフィールド706を介して資源700のタイアを検出する。一旦資源700のヘッダ702が定義されると、資源の中味704の中味は定義しうる。

17.8. 資源エデイタ(第10B図)

第10B図を参照すれば、資源700を作成し、かつ編集するために使用する本発明によるシステムの資源エデイタの構造とオペレーションとが示されている。第10B図には、資源タイアによりグループ化され、かつタイアa資源700a、タイアb資源700b…タイアa資源700nとそれぞれ指示した複数の資源700が示されている。

資源エデイタ722は、実際の資源操作の多くを達成する責任のあるRESPACK機能への呼び出しをよく利用する。その他のプログラムはRESPACK機能への呼び出しにより資源を作成したり修正したりできる。しかしながら、資源エデイタ722は資源を作成したり修正する一次手段である。

18. 資源カストマイゼーション

18.1. カストマイズした資源による一般的なカストマイゼーション

本発明のシステムにおいて、本システムの挙動の多様な局面を、適当な原始資源に対して自動的にカストマイズした資源で代替させる単一の機構を用いて個々のユーザに対してカストマイズできる。

カストマイズ可能特性を有するプログラムを構成する場合、プログラムは各カストマイズ可能特性についてのデータを記憶するために1個以上の資源を使用するようにされる。ユーザが特性を変えると、対応する資源のカストマイズされたバージョンが作成され、そのユーザのユーザプロフィ

(54) 資源エデイタ722は主資源エデイタ724と複数のタイアを特定したサブエデイタ726とから構成されている。各資源タイアにはタイアxのサブエデイタ726があり、第10B図に示すタイアxサブエデイタ726は、例えばタイアaサブエデイタ726a、タイアbサブエデイタ726b、…タイアnサブエデイタ726nに対応するよう指示されている。

図示のように、主資源エデイタ724はユーザ728に対する一次インタフエースを提供し、資源700を作成し、資源700のヘッダ部分702を定義し、編集する装置を含む。主資源エデイタ724は、資源700の中味704を編集し、管理するために直接使用されないが、一旦資源700が作成され、ヘッダ702が定義され、編集されると、タイアxサブエデイタ726の中の対応するサブエデイタを呼び出し、呼び出されたタイアxのサブエデイタ726へのインタフエースを提供する。呼び出されたタイアxサブエデイタ726は次いで、資源700の中味部分704に常駐する資源データを定義し編集するために使用される。

ルに記憶される。その後そのユーザに対してカストマイズした形で特性が現われ、一方他のユーザには元の形で現われる。この作用は、カストマイゼーションを示すプログラムとは独立したメカニズムにより達成される。換言すれば、カストマイゼーションを示すプログラムが、カストマイズした特性を含むオペレーションを実行している場合、そのプログラムがカストマイゼーションがあったか否かについて心配する必要はない。これは、一般的な特性のカストマイゼーションはカストマイズした資源を利用して達成されるからである。RESPACKは自動的に、必要とされた変更のみ実行する、即ちRESPACKは自動的に、適当なときに(即ち、現在のユーザのユーザプロファイルに適当なものが介在するときに)資源のカストマイズしたバージョンと代替する。

RESPACKは特定の資源を(例えばRESget()呼び出しに応答して)検索するとき、特定の資源の資源ＩＤに対応するエントリに対して、現在開放されている資源ファイルで割り出し情報を探索する。

もし現在のカストマイゼーションＩＤ(即ちRESpteastid()を呼び出すことによりセットした)があり、かつ前記エントリにおいてカストマイゼーションフラッグが、資源のカストマイズしたバージョンが介在しうることを示すとすれば、RESPACKはユーザのユーザプロフィルが、特定の資源ＩＤと現在のカストマイゼーションＩＤを備えたカストマイズした資源を含むか否か検出する。RESPACKはもしカストマイズした資源が介在するとすればそれを検索するか、あるいは適当なエントリが見つかった資源ファイルにおいて資源を検索する。

18.2. カストマイゼーションＩＤs

ユーザが資源をカストマイズすると、資源の新しいバージョンが作成される。この資源のカストマイズしたコピイはユーザのユーザプロフィルの一部として記憶される。資源のそれぞれのカストマイズしたコピイはそれにカストマイゼーションＩＤと資源ＩＤとを関連づけている。資源ＩＤはカストマイズされた資源のそれと同じである。カ

のオブジェクトマネージャおよび全てのドキュメントオブジェクトマネージャ)により使用されうる。ユーザはドキュメントオブジェクトマネージャのカストマイゼーションの能力を利用して資源をカストマイズし、資源の修正したコピイを作成する。このカストマイズしたコピイはドキュメントオブジェクトマネージャのカストマイゼーションＩＤ特性と関連する。ドキュメントにオペレートしているときはいつでも、資源のカストマイズしたバージョンが使用される。スプレッドシートに対してオペレートするときはいつでも、スプレッドシートオブジェクトマネージャが別のカストマイゼーションＩＤを使用するであろうから資源の元のバージョンが用いられる。

19. APPACK親玉呼び出し

以下はアプリケーションパック(APPACK)により提供される機能のあるものについての詳細説明を提供する。これは当該技術分野の専門家に「PACK」の概念と用途の双方を示し、かつ本発明のシステムの前述した特徴やオペレーションを更に詳細に

(55) ストマイゼーションＩＤは、ユーザがカストマイゼーションを行ったプログラムにより割り当てられる。

カストマイゼーションＩＤはいずれのカストマイズした資源に対しては一意でない。同じカストマイゼーションＩＤは同じ範囲のカストマイゼーションが好ましいグループの資源に対して用いられる。典型的には、各オブジェクトタイアに単一のカストマイゼーションＩＤが関連している。このように、オブジェクトのあるタイプに対するオブジェクトマネージャの全ては同じユーザのカストマイゼーションを明示する。しかしながら、もし特定のプログラムがその単一のプログラムに対してのみ現われるカストマイゼーションを提供することが好ましい場合、そのプログラムに関連した１個以上の個別のカストマイゼーションＩＤがありうるはずである。

以下の例はカストマイゼーションＩＤの用途を例示する。最初に、資源は多くの種々のオブジェクトマネージャ(例えば全てのスプレッドシート

説明する。

19.1. 呼び出しサービス

19.1.1. APrqstart()スタートリクエストを発す

この機能は所定のリンクにより特定されたアプリケーションを呼び出し、それにSTARTリクエストを送る。このリクエストに対してはリクエストのオプションは何らない。呼び出されたアプリケーションがAPinit()を呼び出すと、リンクと、任意のリンク仕様とを受け取る。この機能は呼び出されたアプリケーションからの応答を期待し、その応答が受け取られると復帰する。それは呼び出されたアプリケーションとは通信をしない。

入力パラメータ

−このリンクに対する親オブジェクトのオブジェクトＩＤへのポインタ

−呼び出すべきオブジェクトへのリンクのＩＤ、もしこの値が零の場合、オブジェクトＩＤへのポインタにより識別されたオブジェクトが直接呼び出される

−リンク仕様の長さ、もし何ら提供されないなら零

−希望に応じリンクを備え、呼び出されたアプリケーションへ送られるべきリンク仕様へのポインタ。これはもしリンクが関連のデータを有するとすればLKFgtspec()を呼び出すことにより得ることができる。もしこのポインタがNULLの場合、仕様は何ら送られない。

出力パラメータ

−無

19.1.2. APrgedit()エデイト リクエストを発す

この機能は所定のリンクにより規定されたアプリケーションを呼び出し、それにEDITリクエストを送る。このリクエストに対しては何らリクエストのオプションはない。呼び出されたアプリケーションがAPinit()を呼び出すと、リンクを任意のリンク仕様とを受け取る。この機能は呼び出されたアプリケーションからの応答を期待し、応答が受け取られると復帰する。それは呼び出された

帰する。それは呼び出されたアプリケーションとは通信を設定しない。

19.1.5. APrgcreate()−作成リクエストを発す

この機能は所定のリンクにより規定されたアプリケーションを呼び出し、それにCREATEリクエストを送る。リンクは新しく作成されたオブジェクトへのものであると想定する。リンクの他に呼び出し元は、サーバアプリケーションから受け取り可能のリンクに対するデータタイプ仕様のリストを提供する必要がある。認定されたリンクデータタイプのみを提供すべきである。呼び出されたアプリケーションがAPinit()を呼び出すと、リンクとデータタイプ仕様とを受け取る。この機能は呼び出されたアプリケーションからの応答を期待し、該アプリケーションとの通信を設定する。この機能は、否定的な答が得られるか、満足に通信が設定されると復帰する。この呼び出しを行った後、アプリケーションはオブジェクト作成プロトコルの残りの部分と共に進行すべきである。

19.1.6. APrgchangelink()−リンク変更リク

(56) アプリケーションとの通信は設定しない。

19.1.3. APrgread()−読出しリクエストを発す

この機能は所定のリンクにより規定されたアプリケーションを呼び出し、それにREADリクエストを送る。このリクエストに対しては何らリクエストのオプションは無い。呼び出されたアプリケーションがAPinit()を呼び出すと、それはリンクとオプションのリンク仕様とを受け取る。この機能は呼び出されたアプリケーションからの応答を期待し、応答が受け取られると復帰する。それは呼び出されたアプリケーションとの通信は設定しない。

19.1.4. APrgrun()−実行リクエストを発す

この機能は所定のリンクにより規定されたアプリケーションを呼び出し、それにRUNリクエストを送る。呼び出されたアプリケーションがAPinit()を呼び出すと、リンクおよび任意のリンク仕様とを受け取る。この機能は呼び出されたアプリケーションから返答を期待し、応答を受け取ると復

エストを発す

この機能は所定のリンクにより規定されたアプリケーションを呼び出し、それにCHANGELINKリクエストを送る。リンクは既存オブジェクトに対するものと想定する。リンクの他に、呼び出し元は、サーバアプリケーションから受け取り可能のリンクに対してデータタイアの仕様のリストを提供する必要がある。認定したリンクデータタイプのみ提供すべきである。呼び出されたアプリケーションはAPinit()を呼び出すと、リンクと、任意のリンク仕様とおよびデータ仕様リストとを受け取る。この機能は呼び出しされたアプリケーションからの応答を期待し、かつ該アプリケーションとの通信を確定する。この機能は、否定的な応答が受けとられるか、通信が首尾よく設定されると復帰する。この呼び出しを行った後、アプリケーションはリンク変更プロトコルの残りの部分と共に進められるべきである。

19.1.7. APrgprint()−プリントリクエストを発す

この機能は所定のリンクにより規定されたアプリケーションを呼び出し、かつそれにPRINTリクエストを送る。リンクの他に、呼び出し元はそのリクエストが実行される要領を示す多数のリクエストのオプションを提供する必要がある。呼び出されたアプリケーションがAPinit()を呼び出すと、リンクと、任意のリンク仕様と、リクエストのオプションとを受け取る。この機能は呼び出されたアプリケーションからの応答を期待する。もしプリントリクエストモードがPRINT INTERACT以外のものであれば通信がなされる。この機能は、プリントリクエストモードに応じて、応答が得られるか、あるいは通信が首尾よく設定されると復帰する。

19.1.8. APrqupdate()－リンク更新リクエストを発す

この機能は所定のリンクにより規定されたアプリケーションを呼び出し、それにLINKUPDATEリクエストを送る。このリクエストに対しては何らリクエストのオプションは無い。呼び出されたア

ジェクトタイアの定義にそのような指示がなされていないとすれば、オブジェクトマネージャが呼び出され実際のコピイオペレーションを実行する。もっともAPrqcopy()の呼び出し元はこの事実に気が付かない。

19.1.10. APrqdeletelink()－リンクを削除する

この機能は特定した親オブジェクトからのリンクを削除する削除後子供のオブジェクトがもはや親のリンクを有していないならば、子供のオブジェクト自体も削除される。オブジェクトタイアの定義にそのように指示されておれば、オブジェクトマネージャが呼び出され実際のオブジェクト削除オペレーションを実行する。呼び出し元はこの事実に気付いていないが。

19.1.11. APrqrelocate()－オブジェクトを再配置する

この機能は所定のリンクにより規定されたオブジェクトあるいはオブジェクトＩＤを特定された物理的位置まで動かす。再配置されたオブジェク

(57) プリケーションがAPinit()を呼び出すと、リンクとリンク仕様とを受け取る。この機能は呼び出されたアプリケーションからの答を期待し、それとの通信を設定する。この機能は、否定的応答が受け取られるか、あるいは通信が首尾よく設定されると復帰する。この呼び出しを行った後、アプリケーションはリンク更新プロトコルの残りの部分と共に進められるべきである。

19.1.9. APrqcopy()－オブジェクトをコピイする

この機能は所定のリンクにより規定されたオブジェクトあるいはオブジェクトＩＤをコピイする。リクエストされるとすれば、新しく作成されたコピイへのリンクが特定された親から設定される。新しいオブジェクトは以下の３個所の中の１個所、即ち呼び出し元により規定された個所、親オブジェクトの近く、あるいは原始オブジェクトの近く、で作成できる。もしオブジェクトが他のオブジェクトに対してリンクを有しておれば、セットコピイ状態フラッグを担持するリンクを有する子供オブジェクトのいずれかもコピイできる。オブ

トまで、あるいはそこからの全てのリンクは、このオブジェクトの他の親や子供が見出せるように調整される。オブジェクトタイア定義にそのように指示されるとオブジェクトマネージャは呼び出され、呼び出し元は気付かないものの実際の再配置オペレーションを実施する。

19.1.12. APrqimport()－外部のオブジェクトをインポートする

この機能はAPPACKオブジェクト環境の外部で作成されたファイル用のオブジェクトカタログにエントリを作成する。この機能は情報をオブジェクトレコードに追加するためにOBJPACKと共に使用しうる有効なOBJIDを戻す。オブジェクトタイアの定義にそのように指示されるとすれば、オブジェクトマネージャが呼び出されて、呼び出し先が気が付かないものの実際のオブジェクトレコード作成オペレーションを実行する。この機能は新しいオブジェクトへのリンクは作成しない。

19.1.13. APinvoke－アプリケーションを呼び出す

この機能は特定のオブジェクトのマネージャを呼び出し所定のリクエストを送る。アプリケーションにはリンク、いずれかの関連リンク仕様および呼び出し者供給のリクエストのオプションが提供される。呼び出し者はまた、再に通信するため呼び出したアプリケーションとの接続が設定されるまで待機を依頼できる。この機能はアプリケーションがスタートされ、リクエストを確認すれば復帰する。

この機能はプログラム間で特殊なインタラクションが要求されるときのみ呼び出されることに注目のこと。その他アプリケーションの呼び出しのほとんどは標準的なリクエスト機能を用いることにより実行すべきである。

入力パラメータ

−このリンクに対する親オブジェクトのオブジェクトＩＤへのポインタ

−呼び出すべきオブジェクトへのリンクのＩＤ、もしこの値が零であるとすれば、オブジェクトＩＤへのポインタにより識別されるオブジェク

トが直接呼び出される

−呼び出されたアプリケーションへ送られるべきリクエストコード。このコードは標準リクエストコードのいずれかとコンフリクトしてはならない。

−応答フラッグ:呼び出しされたアプリケーションから応答が期待されるとTRUE、そうでなければFALSE。もしTRUEであれば、成功か否かは問わず新しくスタートしたアプリケーションから応答が受け取られるまで機能は待機する。

−接続フラッグ:呼び出しされたアプリケーションとの通信が希望されるとすればTRUE、そうでなければFALSE。もしTRUEであれば、接続が設定されるまで、あるいはリクエストに対する否定的応答が受け取られるまで機能は待機する。

−リンク仕様の長さ。何も提供されなければ零

−希望に応じ、リンクと共に呼び出されたアプリケーションへ送るべきリンク仕様へのポインタ。これは、もしリンクが関連のデータを有する

場合LKPgtspec()を呼び出すことにより得られる。もしこのポインタがNULLであれば、何ら仕様は送られない。

−リクエストと共に送るべきリクエストオプションの長さ、何も提供されなければ零。

−リクエストと共に送るべきリクエストオプションへのポインタ。もしこのポインタがNULLであれば、何らオプションは送られない。

−その「環境」のパラメータとして新しいアプリケーションへ渡すべきテキストストリングへのポインタのリストへのポインタ。このリストはNULLのポインタで終るべきである。

出力パラメータ

−このオペレーションの間に更に通信すべく使用するために、APOPID(APPACKオペレーションＩＤ)を受け取る変数に対するポインタ。何ら通信が期待も　望もされないとすれば、このポインタはNULLであるべきである。

19.2. オペレーションサポート

19.2.1. APinit()−アプリケーションリクエスト処理を初期化する

この機能はAPPACK通信を初期化し、そのため呼び出し者はスタートアップリクエストを処理し、マッチメーカオペレーションを実行し、あるいはその他のアプリケーションを開始できる。このAPPACKは他のAPPACKサービスが使用される前に呼び出す必要がある。これは、アプリケーションマネージャにより呼び出されたアプリケーションに送られたスタートアップリクエストを解釈し、中味を呼び出し者へ戻す。この機能はこのリクエストのそれ以上の処理に使用すべきオペレーションＩＤを戻す。もし呼び出し者がアプリケーションマネージャにより呼び出しされてなかったとすれば、戻されるリクエストコードはNONEであり、何らAPOPIDは戻されない。

リクエスト自体の他に、この機能はリクエストに関連したいずれのリンクをも戻し、かつリクエストと共に供給されたリクエスト特定のオプションも戻してしまう。アプリケーションは初期化の間この機能を呼び出すものと想定され、リクエス

トが非インタラクティブ処理に対するものの場合いずれかのユーザには可視のオペレーションを実行する前に機能の呼び出しを行うべきである。

入力パラメータ

－無

出力パラメータ

－呼び出しアプリケーションにより発行されるリクエストコード。もし呼び出しプログラムがアプリケーションマネージャにより呼び出されていたとすれば、この値はNONEである。

－このリクエストに対する応答フラッグ－もし応答が期待されればTRUE、もしそうでなければFALSE。

－このリクエストに対する接続希望フラッグ－もし通信が希望されるならばTRUE、もしそうでなければFALSE。

－このオペレーションの間それ以上のアクティビティに使用するAPOPID。このＩＤは、もし応答を発するならばAPreply()に提供すべきである。

－提供されたOBJIDにより規定されるオブジェ

(59) クトタイプ。もしオブジェクトの名前がリクエストと関連していないならば零。

－リクエストを実行すべきオブジェクトのオブジェクトＩＤ。このフィールドはもしオブジェクトタイプフィールドが零であれば埋められない。

－省略値ヒープから割り当てられ、特定されたオブジェクトを記述するオブジェクトカタログレコードからの属性に関する情報を含むデータのブロックに対するポインタ。このポインタはオブジェクトに関する詳細情報を抽出するためOBJPACKまで通される。もはや必要がなくなれば、このブロックに割り当てたスペースはHPfree()を呼び出すことによりヒープに戻すべきである。NULLのポインタは何らオブジェクトが提供されなかったことを示す。

－アプリケーションを呼び出すことにより送られるリンク仕様の長さ。もしこの値が零であれば、リクエストに伴い何らリンク仕様が送られなかったことを示す。

－省略値ヒープから割り当てられたリンク仕

様に対するポインタ。もはや必要とされなければ、リンク仕様に割り当てられたスペースはHPfree()を呼び出すことによりヒープに戻すべきである。NULLのポインタはリンク仕様が何ら送られなかったことを示す。

－アプリケーションを呼び出すことにより送られたリクエストオプションの長さ。もしこの値が零であれば、リクエストを共に何らオプションが送られていなかった。

－APPACKオプション構造の１つあるいは非識別のバイトのストリングとして、発せられたリクエストに応じてフォーマット化され、省略時ヒープから割り当てられたリクエストオプションに対するポインタ。NULLポインタは、リクエストオプションが何ら特定されなかったことを示す。

19.2.2. APreply()－リクエストに対して応答する

この機能はAPinit()から受け取られたリクエストに対して、あるいは実行中のオペレーションに対して応答を送る。この機能は、呼び出しアプリケーションは応答を待機しているので、APINITの「返答」フィールドがTRUEにセットされると常に呼び出される。APinit()の後呼び出され、応答がSUCCESSでないとすれば、オペレーションは停止され、通信は設定されない。もし返答がSUCCESSであり、呼び出しアプリケーションと呼び出し者の双方が通信を設定したいと所望すれば、使用可能接続が設定される。もし呼び出しアプリケーションあるいは呼び出し者のいずれかが通信を所望しなければ、何ら通信は設定されない。APinit()後以外で呼び出しされるとすれば、「CONNECT」パラメータが否定される。

注:もし応答状態がSUCCESS以外であれば、あるいはもし接続が希望されなければ、オペレーションＩＤはこの呼び出しにより無効とされる。そのような場合に、APopfinish()を呼び出す必要はない。

入力パラメータ

－APinit()またはAPmsreserveにより戻される、応答を送るべきオペレーションのＩＤ。

－リクエストしているアプリケーションに戻すべきエラーコード．あるいはリクエストが受け入れられたとすればSUCCESS．

－接続フラッグ:もし呼び出しアプリケーションとの通信が所望されればTRUE、あるいは何ら接続が必要とされなければFALSE、

出力パラメータ

－無

19.2.3. APevinit()－APPACKイベント仕様を初期化する

この機能は、呼び出し者がAPPACKメッセージを待機できるようUIPACKイベント仕様構造を初期化する．この機能は「マッチ」メッセージ、オペレーションリクエストおよび返答、およびバッファマネジメントイベントとを含む．この機能は構造のアレイの１エレメントを充てんする．呼び出し者はUIVSPEC構造のアレイに対してスペースを割り当てるか、とっておき、かつこれら構造の１個に対するポインタをこの機能に渡す．リストターミネータを含む、アレイの他のエレメントを充てん

クションを支援する高レベルのイベント処理パッケージに提供すべき場合に呼び出すべきである．

入力パラメータ

－APevinit()により準備されていたUIVSPEC構造に対するポインタ

－APPACKイベントに関連すべきアクションコード

－APPACKイベントに関連すべきアクションデータ

出力パラメータ

－無

19.2.5. APevremove()－APPACKイベント仕様を除去する

この機能は、APPACKメッセージを待機するためにイベント仕様エレメントに割り当てられたスペースを空ける．この機能は、もし前に準備されたイベント仕様がそれ以上使用されないと呼び出すべきである．

入力パラメータ

－APevinit()により準備されたUIVSPEC構造

(60) するのは呼び出し者の責任である．この使用を用いて受け取られたイベントは、それらの目的や中味を検出するためAPmsgtest()へ通すべきである．APinit()はこの機能を呼び出す前に首尾よく呼び出されたものと想定する．

入力パラメータ

－無

出力パラメータ

－APPACKメッセージイベントを待機できるようにするに必要な情報で充てんすべきUIVSPEC構造に対するポインタ

19.2.4. APevaction()APPACKイベントに対するアクションコードをセットする

この機能は所定のUIPACKアクションコードとデータとを前に準備されていたAPPACKイベント仕様リストエレメントに関連づける．このエレメントは１個以上のリストエントリを含みうるので、呼び出し者がこれらコードをセットするのは実用的でない．この機能は、APPACKイベントリストエレメントと共にイベント仕様リストが、イベントア

に対するポインタ

出力パラメータ

－無

19.2.6. APmsgtest()－APPACKメッセージイベントをテストする

この機能は提供されたUIPACKイベントレコードをテストして、それが進行しているオペレーションと関連しているAPPACKメッセージイベントであるか否か検出する．もしそうであれば、この機能は関連のメッセージの性質を示すフラッグと、それが属するオペレーションのオペレーションＩＤとを戻す．その結果によって、呼び出し者は多数の他のAPPACK機能の中の１つを呼び出し、メッセージをさらに解釈するものと思われる．

入力パラメータ

－テストすべきイベントレコードに対するポインタ

出力パラメータ

－メッセージイベントが属するオペレーションのＩＤを受け取る変数へのポインタ．この変数

は、もしイベントレコードがAPPACKオペレーションに属していなければNULLにセットされる。

−提供されたメッセージイベントのタイプを示すフラッグを受け取る変数に対するポインタ。以下は可能なタイプのリストである。

＊APPACKマッチメーカのマッチメッセージ−通知されたオペレーションは別のアプリケーションによりマッチングされた。

＊APPACKリクエストメッセージ−メッセージを解釈するためAPmsgrequest()を呼び出す。

＊APPACKオペレーション終了メッセージAPopfinish()を呼び出しオペレーションを終了させる。

＊最近完了した活動オペレーションを取り消すリクエスト−MOVE UNDOプロトコルに従う。

＊新しいオペレーションに対するリクエストを含むAPPACKメッセージ−APmsgnewreq()を呼び出してメッセージを解釈する。

＊データ交換ストリームメッセージ−現在の交換取得機能を呼び出す。

−そのためにRIOIDを戻すべきオペレーションのＩＤ。

出力パラメータ

−このオペレーションと関連したRIOIDを受け取るための変数に対するポインタ

19.2.8. APmsgwait()−APPACKメッセージを待機する

この機能は、特定のオペレーションと関連したイベントを特に待機したいアプリケーションの利益のために提供される。この機能は、このオペレーションと関連した使用可能の接続があるものと想定する。この機能はオペレーションの他方の終りにおいてアプリケーションから到来するメッセージを待機する。呼び出し者は、希望するメッセージが到来する前にAPmsgwait()が戻るようにさせるイベントのリストを供給しうる。さらに、呼び出し者は待機のためのタイムリミットを指定しうる。次いで、イベントレコードポインタが、さらに処理するための(APmsgrequest())のようなAPmsg()解釈機能の１つに通すことができる。

＊オペレーションの他方の終りにおけるアプリケーションにより送られた専用プロトコルメッセージ−APPACKにより解釈されず。

＊APPACKにより扱われる内部コントロールメッセージ−それを否定し、継続する。

＊APPACKでないメッセージ。

19.2.7. APrioinit()−アクチブオペレーションに対するRIOIDを取得する

この機能は、RIOPACKサービス(その上に各種のデータ交換機能が構成される再方向変換可能Ｉ／Ｄ)を用いる標準的なデータ交換サービスのいずれかとのデータ交換を実施するために使用しうるRIOIDを戻す。提供されたRIOIDは特定のオペレーションＩＤと関連した、前に設定された接続に基いている。データ交換が完了すると、呼び出し者は他のAPPACKアクティビティ用のオペレーションＩＤを使用し続けるか、あるいはオペレーションを終了するためにAPopfinish()を呼び出すことができる。

入力パラメータ

入力パラメータ

−メッセージが期待されているオペレーションのＩＤ。

−希望するメッセージが到来する前にこの機能を戻すようにさせるべきイベントを記述した、イベント仕様のアレイに対するポインタ。詳細についてはUIPACK仕様を参照のこと。もしこのポインタがNULLであるとすれば、この機能はメッセージが到来するか、タイムリミットが消滅するまで待機する。

−受け取りの待機を停止する前に待機すべきミリ秒の数、零の値は、この接続のAPPACK省略値が用いられるべきことを示す。WAIT FOREVERの値はその意味するところを意味する。

出力パラメータ

−提供されたメッセージイベントのタイプを示すフラッグを受け取る変数に対するポインタ・メッセージタイプはAPmsgtest()により戻されるものと同じである。

−本機能を早期に戻してしまったイベントレ

コードで充てんすべき構造へのポインタ。このレコードは、希望するメッセージが到来するかあるいは機能がタイムアウトすれば充てんされない。このポインタは、もしイベントレコードが希望でなければNULLでよい。

19.2.9. APosrequest()−オペレーションリクエストを送る

この機能は、呼び出し者が供給したいずれかのリクエストオプションと共に、オペレーション接続の他端においてアプリケーションに特定されたリクエストを送る。他端におけるアプリケーションはAPmsgtest()がタイプAPMSGREQUESTと解釈するメッセージイベントを受け取る。もし応答が希望されると、この機能は、他方のアプリケーションがリクエストを受け取り確認すると復帰する。

入力パラメータ

−リクエストを送るべきオペレーションのＩＤ。

−他のアプリケーションに送るべきリクエストコード。

が処理される前に、送り出されているアプリケーションがこのリクエストに対する応答を期待しているか否か指示する。

入力パラメータ

−APmsgtest()により戻された際、リクエストを適用すべきオペレーションのＩＤ。

−メッセージ用のイベントレコードに対するポインタ。

出力パラメータ

−他方のオペレーションにより送られるリクエストを受け取る変数に対するポインタ。

−他方のオペレーションにより送られる応答フラッグを受け取る変数に対するポインタ。もし応答が期待されるならTRUE、そうでなければFALSEである。

−他方のオペレーションにより送られるリクエストオプションの長さを受け取る変数に対するポインタ。もし戻された長さが零であれば、リクエストと共に何らオプションが送られなかったのである。

(62)

−応答フラッグ:もし応答が呼び出したアプリケーションから期待されるとTRUE、もしそうでなければFALSE。もしTRUEであれば、本機能は成功の如何を問わず新しく開始したアプリケーションから応答が受け取られるまで待機する。

−リクエストと共に送るべきリクエストオプションの長さ。

−リクエストと共に送るべきリクエストオプションを含む構造に対するポインタ。もしこのポインタがNULLであるとすれば、オプションは何ら送られない。

出力パラメータ

−無

19.2.10. APmsgrequest()−リクエストメッセージを解釈する

この機能はタイプAPMSGREQUESTのAPPACKメッセージを解釈し、リクエストについての詳細を戻す。リクエスト自体の他に、この機能はリクエストと共に供給されたいずれかのリクエスト特定のオプションを戻す。応答フラッグは、リクエスト自体

−発せられたリクエストに応じて、APPACKオプション構造の１つとして、あるいは非識別のバイトのストリングとしてフォーマット化された、リクエストオプションに対するポインタを受け取る変数のポインタ。NULLのポインタは何らリクエストが特定されなかったことを示す。

19.2.11. APopfinish()オペレーションを停止する

この機能は進行中のオペレーションを停止させる。この機能は理由を問わず、オペレーションの現在の状態にかかわらずオペレーションを停止すべきときはいつでもこの機能が呼び出される。もしオペレーションが現在２つの接続されたアプリケーションから構成されているとすれば、タイプAPMSGFINISHのメッセージが他方のアプリケーションに送られる。この呼び出し者は停止の理由を指示するエラーコードを供給し、SUCCESSが文字通りの意味であればオペレーションの終りにおいてオペレーションの停止を同意する。もしタイプAPMSGFINISHのAPPACKメッセージが受け取られる

とすれば、他方のアプリケーションの停止信号と、該オペレーションに割り当てられたフリーメモリを取得するためにこの機能も呼び出すべきである。この場合、SUCCESSは状態を示すものとして送るべきである。

入力パラメータ

−停止すべきオペレーションのＩＤ

−他方のオペレーションに送るべき状態のコード

出力パラメータ

−他方のアプリケーションにより送られた停止状態コードを受け取るための変数へのポインタ、この値はもし呼び出し者が最初にオペレーション停止するのであれば、戻されない。

19.3. マッチメーカのオペレーション

19.3.1. APmmreserve()−オペレーションに対するマッチメーカを予約する

この機能は特定のオペレーションに対するマッチメーカの一方の側を予約する。マッチメーカは、オペレーションの半分の細部についてのユーザ仕

＊共用オペレーション

＊マルチコピイオペレーション

＊オブジェクトコンバートアウトオペレーション

＊プレースオペレーション

＊リンクのみ許容したプレースオペレーション

＊リンクを許容していないプレースオペレーション

＊オブジェクトコンバートインオペレーション

出力パラメータ

−所定のオペレーションに対応するマッチメーカの他方の側の現在の状態を示すフラッグを受け入れる変数のポインタ。この変数はマッチメーカを予約できない場合変更されない。

＊他方の側はまだ通知せず

＊他方の側はすでに通知し、待機中

＊他方の側は通知し、リンクのみ期待中

＊他方の側は通知したが、リンクを受け入

(63) 様が進行している間マッチメーカを予約しておくべきである。一旦ユーザ仕様が完了すると、APmmpost()を呼び出すことによりオペレーションに通知すべきである。このオペレーションの他のAPmmreserve()またはAPmmpost()リクエストは、マッチメーカの予約されている間は許容されない。このリクエストを受け入れないようにするオペレーションが現在通知されると、進行中のオペレーションを記述したエラーコードが戻され、そのためユーザに適当に通知できる。もしマッチメーカの状態がAPMMYOURSELFであれば、同じプロセスが通知されたマッチメーカの対応する他方の側を有していることを示す。呼び出し者はこの点でオペレーションを停止するか、APmmpost()を継続しマッチをチェックするか選択できる。

入力パラメータ

−マッチメーカを予約しておくべきオペレーションのコード:

＊コピイオペレーション

＊移動オペレーション

れること不可能

＊他方の側は自身により通知ずみ

−進行中のオペレーションＩＤを受け入れる変数のポインタ、このＩＤはいずれか他の使用がなされる前にAPmmpost()またはAPmmclear()まで送る必要がある。

19.3.2. APmmpost()−マッチメーカに対するオペレーションを通知する

この機能は、マッチメーカに対する進行中の交換オペレーションの半分を通知する。APmmreserve()はそのオペレーションに対してマッチメーカを予約するために先に呼び出しずみとなっている。呼び出し者は、本オペレーションを完成させることに合意する別のアプリケーションに対して提供しうる、あるいはそこから受け取られうるデータ交換のタイプを記載した情報を提供する必要がある。もしこの機能がSUCCESSを戻し、かつ戻されたマッチメーカの状態がAPMMMATCHまたはAPMMYOURSELFであるとすれば、マッチが発生したこと、およびマッチイベントが何ら到来していな

いものと想定しうる。状態がAPMMNOMATCHであるとすれば、呼び出し者はAPopfinish()を呼び出し、ユーザに通知すべきである。

入力パラメータ

－通知すべきオペレーションのＩＤ。

－通知アプリケーションにより提供あるいは受け入れ可能のデータタイプを記述し、APDATALISTENDにセットされた「データタイプ」を備えたエントリにより停止されたAPDATASPEC構造のアレイに対するポインタ。

出力パラメータ

－所定のオペレーションに対応するマッチメーカの他方の現在の状態を示すフラッグを受け入れるポインタへのポインタ。マッチメーカに通知できなければこの変数は変わらない。

＊他方の側はまだ通知されず－待機

＊他方の側は通知されたがマッチメーカは可能でない

＊他方の側は通知され、一致が判明した

＊他方の側は通知され、自身でマッチング

ビスを用いて、データを送ったり、受け取ったりするために使用できる。APopfinish()はこの機能が不首尾のときに呼び出すべきである。さもなければ、接続が設定されるとマッチメーカがクリヤーされる。

入力パラメータ

－接続が希望されるオペレーションのＩＤ

出力パラメータ

－他方のアプリケーションが通知したオペレーションコードを受け取る変数に対するポインタ

－２組織のアプリケーションの間でマッチしたデータタイプの数のカウントを受け取る変数に対するポインタ

－２組織のアプリケーションの間でマッチしたデータタイプを記述し、「データタイプ」をAPDATALISTENDにセットしたエントリにより停止されたAPDATASPEC構造のアレイへのポインタを受け取る変数に対するポインタ、もしポインタがNULLであるとすれば、仕様は何ら戻されない。これ以上必要とされないのであれば、アレイにより

(64) された

－もしマッチメーカの状態がAPMMMATCHまたはAPMMYOURSELFであれば、他方のアプリケーションが通知したオペレーションコードを受け取る変数へのポインタ。

19.3.3. APmmconnect()マッチしたアプリケーションに接続する

この機能は、呼び出し者の通知したオペレーションとマッチしたアプリケーションとの通信を設定する。この機能の呼び出しは、アプリケーションマネージャからAPMSGMATCHメッセージを受け取り、あるいはAPmmpost()からAPMMMATCH状態を受け取った後であるべきである。この機能は他方のアプリケーションにより通知されたオペレーションコード、およびマッチしたデータ仕様のカウントとリストとを戻す。(通信PLACE側にある)コンシューマアプリケーションはこの機能を呼び出した後データに対する特別のリクエストを出すものと思われる。この機能により戻されたAPOPIDはAPPACKリクエスト機能のいずれか、およびデータ交換サー

使用されたスペースをヒープへ戻すようHPfree()を呼び出すべきである。

19.4. リンク交換

19.4.1. LNXpinit()－リンクストリームの構成を開始する

この機能はリンク交換ストリームの構成を開始する。新しいストリームが現在の交換ストリームとなる。一時に使用中のリンクストリームは１個のみしかありえないことに注目のこと。この機能を呼び出した後、他方のLNXp()機能を用いてリンクおよびリンクデータとをストリームに入れることができる。

19.4.2. LNXprestart()－リンクヘストリームをリセットする

この機能はリンク交換データを受け入れるために、特定の接続に対して現在使用中のストリームをリセットする。この機能ははめ込まれたリンクを交換データの別のタイプへ挿入するか、あるいは前のリンクのデータに対して別のデータタイプに切り換えた後リンクを送り始めるために呼び出

しできる．この機能は新しいストリームを作成することができない．ストリームは現在のリンク交換ストリームとなる．この機能を呼び出した後、他のLNXp()機能を用いてリンクとリンクデータとをストリームに投入することができる．

19.4.3. LNXplink()－リンクをストリームに入れる

この機能は既存のリンクを現在のリンクのストリームに入れる．親のオブジェクトは前に定義したリンクを有しており、いずれかの関連したリンク仕様とデータとが前に識別したリンクファイルに記憶されているものと想定される．リンクの他に、該リンクに関連した仕様とデータとがストリームに入れられる．リンクの挿入を完了させるために、他の交換サービスへの呼び出しは必要でない．この機能は多数のリンクを現在のストリームに投入するため繰返し呼び出せばよい．

19.4.4. LNXpnewlink()－新しいリンクをストリームに入れる

この機能は現在のリンクのストリームで新しい

()機能を使用してデータを送り続けるべきである．もしストリームが最初にリンク交換ストリームであった場合、この機能がストリームを完成させ、バッファされたいずれかのデータを接続の他端でコンシューマへ送る．この機能は接続を終了させはしない．

19.4.6. LNXginit()－リンクストリームの読取りを開始する

この機能は新しいリンク交換ストリームの読取りを開始する．新しいストリームが現在交換ストリームとなる．一時に使用中のストリームは１個しかありえないことに注目のこと．リンク交換データは所定の接続上を到来しているものと想定される．この機能を呼び出した後、他方のLNXg()機能を使用してストリームからリンクおよびリンクデータを取得できる．

19.4.7. LNXgrestart()－ストリームをリンクにリセットする

この機能は特定の接続にストリームをリセットしてリセット交換データを読取る．この機能は、

(65) リンクを構成する．呼び出し者は、リンクを設定すべきオブジェクトのＩＤを供給する．受け取る方のアプリケーションは、それがリンク交換情報を受け取るとオブジェクトへリンクを正式に設定する．リンクに関連したリンクデータがあるとすれば、データフラグはセットし、かつ有効なデータタイプとリンク仕様を提供すべきである．次いで呼び出し者は適当な交換サービスを用いてリンクデータをストリームにセットすべきである(XXXprestart()機能への呼び出しから開始する)．LNXprestart()は終了するとリンクストリームへ戻るよう呼び出される．この機能は多数のリンクを現在のストリームに入れるために繰り返し呼び出せばよい．

19.4.5. LNXpfinish()－リンクストリームの構成を終了する

この機能は現在のリンク交換ストリームの構成を終了する．もしリンク交換が別のデータタイプ内にはめ込まれていたとすれば、再びそのデータタイプは現在のデータタイプとなり、XXXprestart

別のタイプの交換データにはめ込まれたリンクを読取り、あるいは前のリンクのデータに対して別のデータタイプに切り換えた後リンクの読取りを再開するよう呼び出すことができる．この機能を呼び出した後、他のLNXg()機能を用いてストリームからリンクおよびリンクデータを読取ることができる．

19.4.8. LNXgLink()－次のリンクをストリームで取得する

この機能は現在のリンク交換ストリームで次のリンクを取得する．この機能は指示された親オブジェクトに対するリンクを作成し、いずれかのリンク仕様と関連のデータとを指示されたリンクファイルに記憶し、新しいリンクを識別すべくリンクＩＤを戻す．リンクに関連したコピイフラッグが指示すれば、オブジェクトはコピイされ、リンクは新しいオブジェクトに対して設定される．新しいリンクはオブジェクトカタログと整合し、そのため親オブジェクトが新しいリンクを担持するものとして整合する．またこの機能はリンクされた

オブジェクトのオブジェクトタイプと、データがリンクに関連しているか否か指示するフラッグと、介在した場合のデータのデータタイプとを戻す。LNKginit()またはLNXgrestart()を呼び出しているときに何らリンクファイルが特定されていなかったとすれば、リンク仕様とデータとは廃却される。データを記憶するためのリンクファイルが全く利用できないとしてもフラッグとデータタイプとは戻される。現在のストリームから多数のリンクを取得するためにこの機能を繰り返し呼び出せばよい。

19.4.9. LNXgpeek()－ストリームで次のリンクをルックアップする

この機能は現在のリンク交換ストリームにおける次のリンクについての情報を戻す。この機能は実際にはリンクを作成しない。この機能は、リンクのオブジェクトタイプあるいはデータタイプをリンクを受け取る前に検査する必要のある場合に使用すべきである。この機能を呼び出した後は、LNXglink()またはLNXgskip()を呼び出してリンク

LNKgpeek()がLNXEENDを戻した後のみ呼び出されるべきであるが、リンク交換ストリームを処理している間はいつでも呼び出すことができる。

19.5. テキストの交換

19.5.1. TXXpinit()－テキストのストリームの構成を開始する

この機能はテキストストリームの構成を開始する。新しいストリームが現在の流れとなる。所定の接続に対して作用するのは１個のストリームのみであることに注目のこと。この機能を呼び出した後、他のTXXp()機能を用いてテキストとテキストデータをストリームに入れることができる。省値がストリームでのヘッダ情報の一部として転送され、かつ後で転送できるように記憶される。

19.5.2. TXXprestart()－ストリームをテキストにリセットする

この機能は特定の接続において現在使用中のストリームをリセットしてテキストを受け入れる。この機能はテキストを別のタイプの交換データへ挿入するか、あるいは前にはめ込んでいたデータ

(66) を処理すべきである。

19.4.10. LNXgskip()－ストリームで次のリンクをスキップする

この機能は現在のリンク交換ストリームで次のリンクをスキップする。LNXgpeek()が丁度リンクを検査するために呼び出され、かつエラーを戻していないものと想定する。この機能を呼び出した後、LNXglink()またはLNXgpeek()のいずれかを呼び出して次のリンクを処理すべきである。

19.4.11. LNXgfinish()－リンクストリームの読取りを終了する

この機能は現在のストリームからのリンク交換データの読取りを終了する。もし別のデータのタイプにリンク交換がはめ込まれていたとすれば、そのデータタイプは再び現在のデータタイプとなり、そのXXXgrestart()機能を用いてデータの読取りを継続すべきである。もしストリームが初めはリンク交換ストリームであったとすれば、この機能はストリームの処理を終了し、それ以上は使用できない。通常この機能は、LNXglink()または

用の別のデータタイプに切り換えた後テキストの転送を再開するために呼び出すことができる。この機能は新しいストリームの作成はできない。ストリームは現在の交換ストリームとなる。この機能を呼び出した後、その他のTXXp()機能を用いてテキストをストリームに入れることができる。

19.5.3. TXXpstring()－テキストストリングをストリームに入れる

この機能はテキストのストリングを現在のテキストストリームへ入れる。このストリングはテキストのレイアウトを制御するためにラインフィードとフォームフィードとを含みうる。これらのキャラクタを備えないストリームは、受け取る方のアプリケーションと文脈とに応じて異なる要領で扱われる。この機能はまた、行間の間隔に明らかに垂直降下移動指令が与えられると有用なキャリッジリターンを含むこともできる。

19.5.4. TXXpchar()－単一キャラクタをストリームへ入れる

この機能は単一のキャラクタを現在のテキスト

のストリームへ入れる。

19.5.5. TXXpfont()－現在のフォントを変える

この機能はセットされたフォント指令を現在のテキストストリームへ挿入し、後続のキャラクタのフォントを変える。もしフォントのidとして0の値が通されると、フォントは伝送の開始時に特定された省略時フォントにセットされる。

19.5.6. TXXpattrs()－テキストの属性をセットする

この機能は使用可能／使用不能属性指令を現在のテキストのストリームに挿入する。この指令は、保持、アンダライニング、ダブルアンダライニングおよびストライクスルーというテキストの属性をセット／リセットするために使用される。これらの値は再びこの指令により変えられるまで有効のままである。

19.5.7. TXXpdiacritic()－区別的発音符を挿入する

この機能は区別的発音符を備えたキャラクタを現在のテキストの流れに挿入するために使用され

に対して、あるいは大きな白紙の部分をスキップするために使用される。この移動は水平位置には影響を与えないため、前のテキストにキャリッジを戻して後続させるべきである。

19.5.11. TXXphorizontal()－水平移動指令を出す

この機能は水平移動指令を現在のテキストストリームへ挿入する。水平移動指令により左右への運動を許容する。ストライクスルーとアンダライン属性とを白紙部分に適用したり(ペンダウンで移動)あるいは適用しない(ペンアップ移動)ようにできる。

19.5.12. TXXpspacing()－行間間隔指令を出す

この機能はセットされた行間間隔指令を現在のテキストストリームへ挿入する。セットされた行間間隔指令が所定のフォントに対して省略時間隔をオーバライドするよう使用される。もし得られた値が正であればその値は自動スペーシングに使用され、もしその値が0であれば、自動スペーシングは何らなされない。もしその値が負であれば、

る。

19.5.8. TXXpstrikethru()－ストライクスルーキャラクタをセットする

この機能はストライクスルーキャラクタ命令を現在のテキストストリームへ挿入する。

19.5.9. TXXpscript()－スクリプトオフセットをセットする

この機能はセットされたスクリプトオフセット指令を現在のテキストストリームへ挿入する。セットされたスクリプトオフセット指令はテキストのベースラインに対する次のスクリプト(スーパ、あるいはサブ)の距離と方向とをセットするために使用される。(スクリプトオフセットをフォントサイズの省略値にセットし戻すために)プリントする際有効である零の値は、テキストの伝送時に通されるとエラーを戻す。

19.5.10. TXXpverfical()－垂直降下移動指令を出す

この機能は垂直降下移動指令を現在のテキストの流れへ挿入する。垂直降下移動指令は行間間隔

省略時値が再び使用される。

19.5.13. TXXplanguage()－言語変更指令を出す

この機能は言語変更指令を入力ストリームへ挿入する。この指令はその後のテキストが異なる言語であることを指示するために使用される。

19.5.14. TXXplink()－リンクをストリームへ入れる

この機能は別のオブジェクトへのリンクを現在のテキストのストリームへ押し始める。この機能を呼び出した後、LNXprestart()を呼び出して他のLNXp()機能への呼び出しを準備すべきである。リンクを挿入することは、リンクの実際のデータを合入するために他のデータ交換タイプも含むことができる。全体のリンクが送られた後、TXXprestart()が呼び出されテキストストリームを再び送り始めるべきである。

19.5.15. TXXpfinish()－テキストストリームの構成を終了する

この機能は現在のストリームへのテキスト交換

データの構成を終了する。もし別のデータタイアにテキスト交換がはめ込まれると、そのデータタイアが再び現在のデータタイプとなり、そのXXXprestart()機能を用いてデータを送り続けるべきである。もしストリームが初めテキスト交換ストリームであったとすれば、この機能はストリームを完成させ、いずれかのバッファしたデータを接続の他端においてコンシューマへ送る。

19.5.16. TXXginit()－テキストストリームの読取りを開始する

この機能はテキスト交換ストリームの読取りを開始する。この機能はストリームの第１の部分が接続の他端から到来し、次いでテキストについての省略時の情報をストリームに戻すのを待機する。新しいストリームは現在の交換ストリームとなる。所定の接続において使用可能の流れは１つのみでありえない。この機能を呼び出した後、他のTXXg()機能が使用され流れからテキストデータを取得するために使用できる。

19.5.17. TXXgrestart()－ストリームをバニ

機能をTXXgrestart()への呼び出しに続いてリンクを取得するよう用いてテキストの取得を続行すべきである。

19.5.19. TXXgstringsize()－次のストリングの長さを取得する

この機能は、ストリームにおいて次にあるテキストのストリングの長さを戻す。この長さを用いてTXXgtext()への呼び出し用バッファを準備すべきである。

19.5.20. TXXgstring()－ストリームからテキストストリングを取得する

この機能は現在のテキストストリームからテキストのストリングを取得する。

19.5.21. TXXgchar()－ストリームからキャラクタを取得する

この機能は入力ストリームから単一のキャラクタを取得する。

19.5.22. TXXgfont()－現在のフォントを取得する

この機能は、入力ストリームにおいて次である、

ラテキストにリセットする

この機能は特定の接続で現在使用中のストリームをリセットしてテキスト交換データを読取る。この機能は別のタイプの交換データにはめ込まれたテキストを読取り、あるいは前にはめ込んでいたデータに対する別のタイプに切り換えた後テキストの読取りを再開するよう呼び出しできる。ストリームは現在の交換ストリームとなる。この機能を呼び出した後、他のTXXg()機能を用いてストリームからテキストデータを読取ることができる。

19.5.18. TXXgnext()－データの次のタイアを入力ストリームで取得する

このルーチンは入力ストリームにおけるデータの次のタイプを検出する。次いで、そのデータを処理するのは呼び出し者の責任である。データを検索することのないこのルーチンへの第２の一連の呼び出しはデータをスキップし、入力ストリームにおける次のデータのタイアを戻す。この機能はデータ指示の終りを戻す。

注：もしタイプがTXXLINKであるとすればLNXg()

セットされたフォント指令に規定されたフォントを戻す。もしセットされたフォント指令が入力ストリームにおける次のデータタイプでないとすれば、最後にセットされたフォント指令に規定された現在のフォントが戻される。もし省略時のフォントがセットされていないとすれば、零フォントが戻される。

19.5.23. TXXgattre()－テキストの属性を取得する

この機能は、入力ストリームにおいて次である使用可能／使用不能属性指令からの属性値を戻す。そのような指令が次でないとすれば、この機能はこれまで出会った最後のそのような指令からの値あるいは省略値(即ち全ての属性が無い)を戻す。

19.5.24. TXXgdiscritic()－区別的発音符を取得する

この機能は、キャラクタと入力ストリームで次にある区別的発音　を取得するために使用される。

19.5.25. TXXgstrikethru()－ストライクスルーキャラクタを取得する

この機能は現在のテキストストリームから新しいストライクスルーキャラクタを取得する。

19.5.26. TXXgscript()－スクリプトオフセットを取得する

この機能は、入力ストリームで次にある、セットされたスクリプトオフセット命令からのスクリプトオフセットの値を戻す。もしそのような指令が次にないとすれば、この機能は最後のそのような指令からの値を戻す。

19.5.27. TXXgvertical()－垂直降下移動指令を取得する

この機能は、入力ストリームにおいて次にある垂直降下移動指令取得する。

19.5.28. TXXghorizontal()－水平移動指令を取得する

この機能は、入力ストリームにおいて次にある水平移動指令を取得する。水平移動指令により左右の移動を許容する。ストライクスルーおよびアンダライン属性を白紙に適用するか(ペンダウンで移動)あるいは適用しない(ペンアップ移動)よ

り、そのXXXgrestart()機能を用いてデータの読取りを継続すべきである。もしそのストリームが頭初からテキスト交換ストリームであったとすれば、この機能はストリームの処理を停止させ、それ以上は使用できない。

19.6. レコードの交換

19.6.1. REXpinit()－バニラレコードストリームの構成を開始する

この機能はバニラレコード交換ストリームの構成を開始する。新しいストリームが現在の交換ストリームとなる。所定のオペレーションには１個のストリームしか使用できないことに注目のこと。この機能を呼び出した後、その他のREXp()機能を用いてレコードをストリームに投入することができる。

19.6.2. REXprestart()－バニラレコードストリームをリセットする

この機能は特定のオペレーションにおいて現在使用中のストリームをリセットするバニラレコード交換データを受け入れる。この機能は、はめ込

まれたレコードを別のタイプの交換データへ挿入するか、あるいは前にはめ込まれたデータに対して別のデータタイプに切り換えた後レコードを再び送るために呼び出すことができる。この機能を呼び出した後、他のREXp()機能を用いてレコードをストリームに投入することができる。

19.6.3. REXpheader()－ヘッダレコードを構成する

この機能はヘッダレコードを現在の交換ストリームへ構成する。レコードは入力パラメータ値に基き構成される。不明の値はNULLとして通過させるべきである。

サブタイアフィールドはアプリケーションに柔軟性をもたせるためのものである。もしアプリケーションがサブタイプ修飾子付のレコードを生成させる意図の場合、サブタイプidsをAPPACKに整合させるべきである。このためサブタイアフィールドは全てのアプリケーションを通して一定のままとしうる。

19.6.4. REXprinit()－レコード記述子を開

始する

この機能は現在の交換システムにおいてレコード記述子を開始させる。この機能はレコード記述子当り１回呼び出される。

サブタイアのフィールドはアプリケーションに柔軟性を提供するためのものである。アプリケーションがサブタイア修飾子を備えたレコードを生成させる意図の場合サブタイアidsをAPPACKに整合させるべきである。このためサブタイアフィールドは全てのアプリケーションを通して一意のままとしうる。

19.6.5. REXpfdesc()－フィールド記述子を構成する

この機能は現在の交換ストリームにおいてフィールド記述子を構成する。この機能はレコード記述子の各フィールドに対して１回呼び出すべきである。

19.6.6. REXprfini()－レコード記述子を終了する

この機能は完全なレコード記述子の終りをマー

19.6.9. REXpfini()－データレコードを終了する

この機能は完全なデータレコードの終りをマークする。この機能はデータレコード当り１回呼び出される。

19.6.10. REXpfinish()－バニラレコードストリームの構成を終了する

この機能は現在のストリームにおけるバニラレコード交換データの構成を終了する。もしバニラレコード交換が別のデータタイプ内にはめ込まれたとすれば、そのデータタイプは再び現在のデータタイプとなり、そのAPXXptreset()機能を用いてデータを送り続けるべきである。もしストリームが頭初からバニラレコード交換ストリームであったとすれば、この機能はストリームを完成させ、バッファされたいずれかのデータをオペレーションの他方の終りにおいてコンシューマに送る。次いでストリームが　了し、それ以上使用されない。

19.6.11. REXginit()－バニラレコードストリームの読取りを開始する

(70) クする。本機能はレコード記述子当り１回呼び出される。

19.6.7. REXpdinit()－データレコードを開始する

この機能はデータレコードを初期化する。この機能はデータレコード当り１回呼び出される。

サブタイアフィールドはアプリケーションに柔軟性を提供するためのものである。もしアプリケーションがサブタイア修飾子付のレコードを生成させる意図であれば、サブタイアidsをAPPACKに整合させるべきである。このため全てのアプリケーションを通してサブタイアフィールドが一意のままとしうる。

19.6.8. REXpdata()－データフィールドを構成する

この機能は現在の交換システムにおいてデータフィールドを構成する。入力データタイアは対応する標準的な交換フォーマットに変換される。この機能はデータレコードの各フィールド当り１回呼び出される。

この機能はバニラレコード交換ストリームの読取りを開始する。この機能はストリームの最初の部分がオペレーションの他端においてサーバから到来するのを待機する。新しいストリームが現在の交換ストリームとなる。所定のオペレーションで使用できる流れは１つのみであることを注目のこと。バニラレコード交換データのリクエストがすでにサーバへ送られており、満足な応答が受け取られたものと想定する。この機能を呼び出した後、他のREXg()機能を用いてストリームからレコードを取得するよう使用できる。

19.6.12. REXgrestart()－ストリームをバニラレコードにリセットする

この機能は特定のオペレーションにおいて現在使用中のストリップをリセットしてバニラレコード交換データを読取る。この機能は、はめ込まれたレコードを別のタイアの交換データから読取るか、あるいは前にはめ込んだデータに対して別のデータタイプに切り換えた後レコードの読取りを再開するために呼び出すことができる。このスト

リームは現在の交換ストリームとなる。この機能を呼び出した後、他のREXg()機能を用いてストリームからレコードを読取ることができる。

19.6.13. REXgtype()−次のレコードタイプを取得する

この機能はレコードタイプの値とサブタイプとを戻す。どのタイプのレコードが後続するかを検出し、それを正確に処理するのはコンシューマの責任である。データ転送の終りはエラーとしてこの機能により戻されることに注目することが重要である。

19.6.14. REXgheader()−ヘッダレコード情報を取得する

この機能は現在の交換ストリームにおいてヘッダレコード情報を戻す。

19.6.15. REXgfdesc()−次のフィールド記述子を取得する

この機能は次の順のフィールド記述子を戻す。レコード記述子の終りはこの機能からエラーとして戻されることを注目することが重要である。

(71) フィールド記述子をデータタイプに基いて正確に処理することはコンシューマの責任である。

19.6.16. REXgdata()−次のデータフィールドを取得する

この機能はフィールドデータとタイプとを戻す。データレコードの終りはこの機能から戻されることを注目することが重要である。どのタイプのデータを処理すべきか検出することはコンシューマの責任である。タイプ依存型情報についてはAPRXFDATA構造の定義を参照のこと。またバニラレコードデータ構造を参照。

19.6.17. REXgnext()−次のレコードタイプにスキップする

この機能は所定タイプの次のレコード、レコード記述子あるいはデータレコードを見付ける。タイプはユーザが指定する。

19.6.18. REXgfinish()−バニラレコードストリームの読取りを終了する

この機能は現在のストリームからバニラレコード交換データの読取りを終了する。もしバニラレ

コード変換が別のデータタイプにはめ込まれていたとすれば、そのデータタイプが再び現在のデータタイプとなり、そのAPXXgreset()機能を用いてデータの読取りを継続すべきである。もしストリームが頭初バニラレコード交換ストリームであったとすれば、この機能はストリームの処理を停止し、それ以上は使用されない。

20. RESPACK機能呼び出し

20.1. 資源ファイルアクセス機能

20.1.1. RESfopen()−資源ファイルを開ける

この機能は読出し専用アクセスのために既存の資源ファイルを開ける。アプリケーションは資源ファイルを何枚開けてもよい。

20.1.2. RESfinish()−資源ファイルのリストを開ける

この機能は呼び出し者が特定する資源ファイルのリストを開ける。各ファイルの名称は呼び出し者が選したストリング、プログラムの省略時言語コード、およびプログラム自体がそこから来たカタログあるいはライブラリからの情報の径路から

構成される。ファイルは読出し専用アクセスに対して開かれる。

この機能に通されたパラメータリストは不変であって、実行すべきアクションによって決まる。第1のパラメータは常にインターナショナルな位置コードである。これに続き、多数のフラッグ／ストリングのペアがあり、ここでフラッグが開くべきファイルのタイプ(システム資源ファイル、言語依存型アプリケーション資源ファイルあるいは言語とは独立したアプリケーション資源ファイル)を特定し、ストリングが開けるべきファイルの名前または接頭語を特定する。あるフラッグの値は後続のストリングポインタを必要としない。パラメータリストは常にRESFNENDのフラッグと共に終了する。

パラメータストリングのフォーマットとファイル名称を構成する方法とはシステムによって決まる。この機能は開かれたファイルのファイルＩＤは戻さない。プログラムは　に各資源に対して入手可能の全ての資源ファイルを通して探索するも

のと想定される。もしファイルの１個を開ける際エラーを経験すると、その他全てのファイルも閉かれ、エラーは戻される。

20.1.3. RESfclose()−資源ファイルを閉じる

この機能はRESfopen()により開かれた資源ファイルを閉じる。

20.2. 資源アクセス機能

20.2.1. RESget()−資源を取得する

この機能はユーザのファイルあるいは資源ファイルにおいて資源を見つける。タイアがマッチすれば、この機能は資源用の省略時ヒープからのスペースを割り当て、メモリへ読取り、ポインタをその資源のメモリコピイまで戻す。タイプがマッチしなければエラーが戻される。

タイプパラメータがRESTANYであるとすれば、いずれの資源タイアともマッチし、ミスマッチエラーは何ら戻されない。

成功すれば、資源はユーザのプロフィルあるいは、もしユーザプロフィルで見つからなければ開かれた資源ファイルの１個から読取られる。呼び

エラーが戻されるが、その他全ての資源はいずれにしても読出しされる。もしRESEOUNDEFINED以外の何らかのエラーが読取りの間に発生すれば、全てのポインタはNULLにセットされ、全ての割り当てられたスペースが自由とされ、エラーが戻される。

資源がもはや必要とされないと、RESrelease()を読取られた各資源に対して呼び出しそれにより占有されていたスペースを割り当て直すようにすべきである。

入力パラメータ

−検索すべき資源ＩＤのアレイの始まり

−検索すべき資源の数

出力パラメータ

−資源へのポインタが記憶されるアレイの始まり

20.2.3. RESpoint()−資源へのポインタを取得する

この機能は資源の一部へのポインタは戻す。この機能は、資源が潜在的に極めて大きく、かつア

(72) 出し者が資源を見つけると、本機能はRESrelease()を呼び出すことにより解放され、そのためヒープの前記スペースは再生利用しうる。

入力パラメータ

−検索すべき資源のＩＤ

−実際の資源タイアを妥当化すべき資源の期待されるタイアあるいはRESTANY

出力パラメータ

−資源に対するポインタが記憶されている領域へのポインタ

−資源のサイズが記憶されている変数へのポインタ。もしNULLであれば、サイズは戻されない。

20.2.2. RESmget()−多数資源を取得する

この機能は多数の資源を省略時のヒープへ読取り、それらのポインタを戻す。ある環境下では、この機能はアクセスすべき各資源に対して、RESget()を呼び出すことよりも顕著な性能上の利点を提供しうる。タイプの妥当性が得られないと資源のサイズは戻されない。もし資源が見つからなければ、資源へのポインタはNULLにセットされ、

アリケーションがそれをバッファにコピイするというオーバヘッドを排除したいと希望するときに呼び出すべきである。資源の断片はアプリケーションに提供しうるが、修正はできない。資源を修正しようと試みても予想の出来ない結果をもたらす。資源がもはや必要とされなくなれば、RESrelease()を呼び出すべきである。

一時にマッピング可能な資源の断片の数は極限定されている。最大数の断片数に達すると、既存の断片に対してRESrelease()を呼び出すことなく別の断片のポインタを得ようとしてもエラーを発生させる。

特定されたオフセットが資源サイズより大きい場合、エラーは戻される。長さをプラスした特定オフセットが資源のサイズより大きいならば、異なるエラーが戻され、資源はいずれかにマッピングされる。

入力パラメータ

−検索すべき資源のＩＤ

−検索すべき資源内のオフセット

−必要な資源データの量(バイト)

出力パラメータ

−資源へのポインタが記憶される領域へのポインタ

20.2.4. RESrelease()−資源を解放する

この機能はメモリでの資源のコピイを解放することによって、必要ならスペースを再生利用できる。この機能はHPfree()直接呼び出しでなくむしろ常に利用すべきである。

入力パラメータ

−RESget()、RESaget()あるいはRESpoint()により戻される資源データポインタ

出力パラメータ

−無

20.2.5. RESread()−資源を読取る

この機能は資源の全部または一部をユーザが特定の領域へ読取る。もし提供されたバッファが要求された資源のサイズより大きいか、あるいは資源がバッファより大きければエラーが戻されるが、資源は依然としてバッファに置かれる。

ユーザが特定した情報構造は要求された情報と共に充てんされるユーザの変数に対するポインタを含むべきである。もし前記構造におけるフィールドがNULLの場合、その情報は戻されない。さもなければ情報はフィールドにより指摘された変数にセットされる。もし名称、記述あるいはリテラルフィールドがセットされるとすれば、適当なテキストストリングが省略時のヒープに読取られ、常で終り、呼び出し者のポインタはそこを指すようセットされる。この場合、呼び出し者は、ストリングがもはや必要とされないならばHPfree()で割り当てられたスペースを解放する必要がある。そのようなストリングの無い場合、呼び出し者のポインタはNULLにセットされる。

20.3. 資源ファイル管理機能

20.3.1. RESfcreate()−資源ファイルを作成する

この機能は新しい資源ファイルを作成し、そのファイルに対して編集オペレーションが実行できるようにする。minidおよびmaxidがファイルで作

(73) 入力パラメータ

−検索すべき資源のＩＤ

−読取りされている資源内のオフセット

−読取るべき資源のバイト数

出力パラメータ

−資源データが記憶される領域へのポインタ

20.2.6. RESlookup()−所定の名称で資源を見つける

この機能は、特定の名称を有する資源のＩＤと、その中で資源が発見されたファイルのＩＤとを戻す。その名前の資源がない場合、エラーが戻される。特定の名前で資源が見つかるが、より最近に開かれたファイルには同じ資源ＩＤを有する別の資源がある場合、資源とファイルのＩＤがセットされるが、エラーは依然として戻される。この理由は、資源ＩＤを資源のアクセスに使えないからである。

20.2.7. RESgtinfo()−資源に関する情報を取得する

この機能は資源に関する情報を戻す。

成しうる最小と最大の資源ＩＤsを定義する。この範囲より外のＩＤでファイルに資源を作成しようとしてもエラーを戻すことになる。一旦ファイルが作成されると、これらの値は決して変更できない。

20.3.2. RESfedit()−資源ファイルを修正する

この機能は資源エディタが既存の資源ファイルの中味を修正できるようにする。この機能は該ファイルに対する最低および最高の合法的な資源ＩＤを戻す。

20.3.3. RESfview()−資源ファイルを検査する

この機能は資源エディタがユーザファイルから干渉されることなく、RESgtnext()、およびRESgtprev()による走査を含み、資源ファイルの中味を観察できるようにする。ファイルは読出し専用に開かれているので、資源は何ら修正されないが、編集されたファイルに適用されるその他の機能のいずれも使用可能である。

20.3.4. RESfcommit()−資源ファイルの修正を行う

この機能は、RESEFIDを戻す機能のいずれかにより開けられた資源ファイルへの修正を完了する。もしファイルがユーザプロファイルである場合、それも閉じられる。

資源ファイルになされた変更はファイルのパーマネントコピイに組み込むことができ、あるいは変更を捨てファイルを変えないままとしうる。もしファイルがRESfedit()により開けられ、非変更パラメータがTRUEであれば、ファイルに対してなされた全ての変更は棄却される。もしファイルがRESfcreate()を用いて開けられたとすれば、それも削除される。もしパラメータがFALSEであれば、変更はファイルのパーマネントコピイへ組み込まれる。

このルーチンに通されて来た資源ファイルＩＤがRESmapedit()を介して作成されたとすれば、ファイルＩＤは削除されるが、関連のプロファイルは影響されない。呼び出し者は依然としてプロファイルを閉じる責任がある。

20.3.5. RESgtfinfo()−ファイルに関する情報を取得

この機能は、編集のために開けてある資源ファイルについての数個の情報、即ちファイルにおける資源の全数、現在の最小および最高の資源ＩＤ、および外部および内部のファイルバージョン番号とを戻す。これらの番号は、現在の編集セッションの間作成されたり、かつ／または削除された資源の数を考慮する。もしピクチャのいずれかがNULLであるとすれば、相応の数の情報は戻されない。

最低および最高の既存の資源IDは、RESfcreate()により受け入れられ、RESfedit()により戻された、それぞれファイルにおける最低および最高の資源ＩＤであるＩＤとは同じでない。

20.3.6. RESptfinfo()−ファイルに関する情報を入れる

この機能は、資源ファイルに記憶可能の最小と最大の資源ＩＤを修正する。もしその範囲が違法値を含んでいるか、あるいはすでにファイルに存在している資源とを除外しているとすればエラー

が戻される。

20.3.7. RESavail()−未使用の資源ＩＤを取得する

この機能は、すでにある資源により使用されていない、ある特定範囲内の、ファイル内の最小または最高資源を戻す。この機能は、システムが生成の資源ＩＤを選択する上で資源エディタにより使用される。

20.3.8. RESgtnext()−次の資源情報を取得する

この機能は、特定のファイルの特定の資源の後で物理的に位置した資源に関する情報を戻す。その引き数はRESIDBOFとなり、その場合そのファイルの最初の資源は戻される。もし特定の資源がファイルにおける最後の資源である場合、エラーは戻される。

20.3.9. RESgtprev()−以前の資源情報を取得する

この機能は、特定ファイルの特定資源の直前の資源に関する情報を戻す。引き数はRESIDEOFでよく、その場合ファイルの最後の資源が戻される。特定の資源がファイルでの最初の資源である場合、エラーが戻される。

20.3.10. RESchkpt()−資源ファイル更新のチェックポイント

この機能は、何らかのそれ以上の修正がなされる前に、特定の資源ファイルの現在の状態を記録する。もしRESrevert()が呼び出されるとすれば、RESchkpt()およびRESrevert()との間で実施された全ての修正が棄却される。このオペレーションは、このオペレーションの作用を要するシステムに対する「取り消し」オペレーションをサポートする。

チェックポイントは１レベルのみ可能である。一旦この機能が呼び出されると、この機能に対する２回目の呼び出しにより最初の呼び出しにより記憶されたデータを忘れさせる。これらの機能の呼び出し者はどの程度の細分性の「取り消し」支援を提供すべきか、即ち何が全体的に取り消す必要のある単一のユーザオペレーションを構成するか

を自分自身で決定する必要がある。

20.3.11. RESrevert()−最後のチェックポイント

この機能により、特定の資源ファイルを、それが最後のRESchkptRpt()としておかれていたのと同じ状態に戻されるようにする。その時点以来なされた全ての編集上の修正は廃却する。

20.3.12. RESfreeze()−資源ファイルバージョンを凍結する

この機能は資源ファイルを修正して解放および分配の準備をさせる。もしnewverフラッグがセットされると、外部のバージョン番号が増分される。もしclralterフラッグがセットされるとすれば、(現在の編集セッションの間修正された資源を含む)全ての資源のRESPALTEREDフラッグがクリヤされる。もしstriplitフラッグがセットされるとすれば、全てのリテラルはサイズを小さくするために資源ファイルから除去される。

20.3.13. RESgtfver()−資源ファイルバージョン番号を取得する

エラーは戻されるが、資源は依然としてバッファに位置される。

20.4.2. RESgtcur()−現在の資源情報を取得する

この機能は、資源の情報になされたバージョンについての情報を戻すRESgtinfo()とは対照的に、資源の現在の状態についての情報を戻す。

呼び出し者は、資源に対しどのファイルを探索すべきか規定するか、編集のために用いている全てのファイル内での探索を規定しうる。この機能は、ファイルの間で資源の探索を行いうる唯一の編集機能である。その他の全ての編集機能に対して探索すべきファイルは明確に特定する必要がある。

情報構造は、要求された情報で充てんされるユーザ変数に対するポインタを含むべきである。前記情報構造におけるいずれかのフィールドがNULLであるとすれば、その情報は戻されない。さもなければ、情報はフィールドにより指摘された変数へセットされる。もし名称、記述あるいはリテラ

(75) この機能は資源ファイルに記憶されたバージョン番号のストリングを検索する。このバージョン番号は8キャラクタのフラッグでなければならない。資源ファイルがRESPACKにより作成されると、バージョン番号は「00.00.00」にセットされるが、いつでもRESptfver()を用いて変更しうる。

20.3.14. RESptfver()−資源ファイルバージョン番号を入れる

この機能は8キャラクタのバージョン番号ストリングを資源ファイルにセットする。このストリングはRESgtfver()により検索できる。

20.4. 資源編集機能

20.4.1. RESrdcur()−資源の現在のバージョンを読取る

この機能は特定の資源ファイルから、資源の現在の値を読取る。この機能は、バッファの未編集バージョンのみを戻すRESread()と対照的である。資源はユーザが提供したバッファへ読取られる。もし提供されたバッファが要求された資源より大きいか、あるいは資源がバッファより大きければ、

ルフィールドがセットされるとすれば、適当なテキストストリングが省略時ヒープへ読取られ、かつNULLで終るようにされ、呼び出し者のポインタは該ストリングにセットされる。この場合、呼び出し者は、ストリングがもはや必要とされなければHPfree()で割り当てたスペースを解放する必要がある。そのようなストリングがないとすれば、呼び出し者のポインタはNULLにセットされる。

20.4.3. REScreate()−資源を作成する

この機能は特定ファイルにおいて新しい資源を作成する。資源は特定のパラメータで作成され、データは何ら含まない。もし資源が非零サイズで作成されるとすれば、そのデータ領域は零で充てんされる。資源はファイルにおけるいずれかの希望する相対位置で作成できる。

これは新しい資源を作成しうる唯一の機能である。

情報構造は新しい資源の属性に対するある値を含む。タイプおよびフラッグフィールドは情報構造内で特定せねばならず、あるいはエラーが戻さ

れる．全ての他のフィールドは任意である．もしバージョンが特定されないとすれば、資源バージョンが内部資源ファイルバージョンにセットされる．

20.4.4. RESwrite()－資源を書込む

この機能はユーザにより提供されたデータを特定ファイルの設定資源に書込む．資源にすでに介在している必要がある．資源にすでにデータが介在しているとすれば、古いデータを捨て、資源サイズが特定のカウントにセットされる．

20.4.5. RESrewrite()－資源を重ね書きする

この機能はユーザにより提供されたデータを特定ファイルの既存資源に書込む．これは、すでに資源にあるデータに追加するか部分的に重ね書きできる．

書込みが始まるオフセットは資源の現在のサイズより小さいか、あるいは等しくあらねばならない．いずれのより大きい値もエラーを戻すようにさせる．特定のオフセットプラス長さが現在の資源サイズより大きいとすれば、このサイズは新しいデータを収容するために増大する．

イルと同じ情報およびデータで宛て先ファイルにおいて作成される．コンフリクトがあれば、宛て先ファイルにすでにある資源が最初に取り消される．もしインデックスがコピイされつつあるとすれば、呼び出し側は古いインデックスを保持すべきか、壊すべきかあるいは古いインデックスと新しいインデックスの中味を併合すべきか指定するオプションを有する．

この機能はそれ以上の構成を必要としない．

20.4.10. RESdelete()－資源を削除する

この機能はファイルから資源を削除する．

20.5. 資源索引機能

資源インデックスは、任意的に分類され、全て単一の資源ＩＤにより識別された単一の資源に記憶された均一サイズのエントリの単純なリストである．インデックスエントリは３種類の情報、即ちキー、データの任意チャンクおよび資源ＩＤとを含む．キーが与えられると、呼び出し者は資源インデックスにおけるそのキーでエントリを探し、その結果そのキーと関連したデータと資源ＩＤと

20.4.6. RESptinfo()－資源に関する情報を入れる

この機能は資源に関する情報をセットする．もし資源のサイズが増えると、新しい資源スペースは零で充てんされる．

この情報構造はセットすべき値へのポインタを含むべきである．情報構造におけるいずれのフィールドもNULLであるとすれば、資源の対応する属性は不変のままとされる．フラグの中のRESFNAMEDフラッグが使用されず、情報構造に記憶された名前ポインタから再び取り出される．

20.4.7. RESmove()－資源を新しい位置へ移す

この機能は既存の資源を資源ファイルの新しい位置まで移す．

20.4.8. RESptnum()－資源の番号を付け直す

この機能は特定の資源の資源ＩＤを変える．

20.4.9. RESmerge()－資源リストをファイルに併合する

この機能は一方のファイルから他方のファイルへ資源のリストを併合する．資源は通常原始ファ

を受け取ることができる．もしインデックスが分類されるとすれば、全てのエントリはそれらのキーの値の順に配列される．もしエントリの資源ＩＤが別の資源インデックスのＩＤと定義されるとすれば、インデックス構成者は多くのレベルの階層の関連資源インデックスを作成することができる．

20.5.1. RESixinit()－資源インデックスの構成を開始する

この機能は資源インデックスの構成過程を初期化させる．呼び出し者はキータイプを特定し、インデックスの挙動を分類して、資源インデックスが構成されるにつれて資源を識別するために使用すべきＩＤを受け取る．この機能は資源を資源ファイルに入れない．呼び出し側は資源のファイルへの投入を別のオペレーションにおいて行う必要があり、そうする前にRESixfini()を呼び出す必要がある．

minid、maxidおよびrestypeパラメータが割り出すべき資源に関する情報を提供する．これらの

フィールドはRESPACKによっては全く使用されず、呼び出し側の便宜上インデックスに関する情報を記憶するためのものである。

20.5.2. RESixprep()−既存インデックスの修正を開始する

この機能は、ファイルから読出されるにつれて資源インデックスへのポインタを取り、その修正可能コピイへのポインタを戻す。この機能は他のRESix()機能のいずれかが呼び出される前に行うべきである。いずれにしても入力ポインタは修正されない。もし(例えばRESget()により)ヒープから割り当てられたとすれば、呼び出し側はそれを解放すべきである。

20.5.3. RESixadd()−資源インデックスエントリを追加する

この機能は特定のキー、データおよび資源ＩＤで新しい資源インデックスエントリを構成し、所定の資源インデックスへ挿入する。分類したリストに対して、エントリは照合順序内の適正位置で挿入される。未分類のリストに対しては、エント

(77) リはリストの終りに付属される。提供されたRESIXIDはRESixinit()またはRESixprep()への呼び出しにより戻されたものと想定する。

20.5.4. RESixdelete()−資源インデックスエントリを削除する

この機能は資源インデックスから特定のエントリを削除する。提供された資源ＩＤはRESixnit()またはRESixprep()の呼び出しにより戻されたものと想定する。

20.5.5. RESixfini()−資源インデックスの構成を終了する

この機能は資源インデックスの構成過程を完了させ、かつ資源ファイルに追加可能の資源を戻す。この機能は実際にはRESixprep()により実行されたオペレーションの逆である。戻された資源はヒープから割り当てられ、そのため資源ファイルに挿入された後、EPfree()を呼び出すことにより解放すべきである。

20.5.6. RESixlookup()−資源インデックスエントリをルックアップする

この機能は特定の資源インデックスを取得し、所定のキーで該インデックスを通してエントリを探索し、そのキーに関連したデータと、資源ＩＤとを戻す。この機能は探索を終った後資源インデックスを解放する。もし単一の資源インデックスを通してのマルチパスが必要であれば、RESget()、RESixprep()およびRESixget()を用いる方が速くなる。

20.5.7. RESixulookup()−USHORTエントリをルックアップする

この機能は特定の資源インデックスを取得し、所定のキーで、該インデックスを通してエントリを探索し、そのキーに関連した資源ＩＤを戻す。この機能はUSHORTのキータイプを備えた資源インデックスに対してのみ使用可能である。インデックスデータはこの機能によって戻されないので、エントリデータを取得するにはRESixget()を使用のこと。この機能により探索を終った後資源インデックスを解放させる。単一の資源インデックスを通してのマルチパスが必要であるとすれば、

RESget()およびRESixufind()を用いる方が速い。

20.5.8. RESixllookup()−ULONGエントリをルックアップする

この機能は特定の資源インデックスを取得し、所定のキーで該インデックスの中でエントリを探索し、そのキーに関連した資源ＩＤを戻す。この機能はULONGのキータイプを備えた資源インデックスに対してのみ使用可能である。インデックスエントリはこの機能によって戻されず、エントリデータを取得するにはRESixget()を用いること。この機能は探索終了後資源インデックスを解放する。もし単一の資源インデックスを通してのマルチパスが必要であるとすれば、RESget()およびRESixfind()を用いる方が速い。

20.5.9. RESixslookup()−ストリングエントリをルックアップする

この機能は特定の資源インデックスを取得し、特定のキーで該インデックスの中でエントリを探索し、該キーに関連した資源ＩＤを戻す。この機能はストリングのキータイプを備えた資源イン

デックスに対してのみ使用可能である．インデックスエントリデータはこの機能によって戻されず、エントリデータを取得するにはRESixget()を用いること．この機能は探索終了後、資源インデックスを解放する．もし単一の資源インデックスを通してのマルチパスが必要であるとすれば、RESget()およびRESixnfind()を用いる方が速い．

20.5.10. RESixnfind()－USHORTエントリをルックアップする

この機能は所定のキーで所定の資源インデックスのエントリを探索し、そのキーに関連した資源ＩＤを戻す．この機能はUSHORTのキータイアを備えた資源インデックスに対してのみ使用可能である．インデックスエントリデータはこの機能によって戻されないので、エントリデータを取得するにはRESixget()を用いること．

20.5.11. RESixlfind()－ULONGエントリを見出す

この機能は所定のキーで所定の資源インデックスのエントリを探索し、そのキーに関連した資源

でエントリを取得する

この機能はインデックス内のその位置により特定された、資源インデックスのエントリを戻す．もし特定のエントリが存在していないとすれば、何ら情報は戻されない．

20.5.15. RESixinfo()－資源インデックスに関する情報を取得する

この機能は資源インデックスを記述した情報を戻す．資源インデックスは資源アクセス機能の１つを呼び出すことによりすでに集められているものとする．

20.6. バッチスタイルの資源作成機能

このセクションはバッチスタイルの資源ファイル構成に使用する一連の資源ファイル作成機能を記述する．これらの機能は資源ファイルを作成するより単純な手段を提供する．但しファイル構成プロセスは非インタラクチブスクリプトから推進されるものとする．既存のファイルを編集したり、あるいはすでに資源ファイルに入れられた資源を修正する機能は何ら提供されていない．

(78) ＩＤを戻す．この機能はULONGのキータイアを備えた資源インデックスに対してのみ使用可能である．インデックスエントリデータはこの機能によって戻されず、エントリデータを取得するにはRESixget()を用いること．

20.5.12. RESixfind()－ストリングエントリを見出す

この機能は所定のキーで所定の資源インデックスのエントリを探索し、そのキーに関連した資源ＩＤを戻す．この機能はストリングのキータイアを備えた資源インデックスのみに対して使用可能である．インデックスエントリデータはこの機能によって戻されず、エントリデータを取得するにはRESixget()を用いること．

20.5.13. RESixget()－資源インデックスエントリを取得する

この機能は所定のキーで所定の資源インデックスのエントリを探索し、そのキーに関連したデータと資源ＩＤを戻す．

20.5.14. RESixindex()－資源インデックス

20.6.1. REScreate()－バッチスタイルの資源ファイルを作成する

この機能は新しい資源ファイルを作成し、資源を挿入すべく準備する．このファイルは既存のものではない．資源の範囲は定義した限度内に入る必要がある．

20.6.2. RESadd()－資源ファイルに資源を追加する

この機能は所定の資源を現在開いている資源ファイルに追加する．資源のデータ部分は前に挿入された資源のすぐ後に挿入し、所定の資源ＩＤを関連させる．この機能により挿入された全ての資源は「可視」「修正可」および「カストマイズ可」とフラッグされる．

20.6.3. RESclose()－資源ファイルの構成を終了する

この機能は現在の資源ファイルの構成を終了し、該ファイルを閉じ　．

20.7. ユーザプロファイルサポート機能

このセクションはユーザプロファイルとRESPACK

を介して対話するよう機能する．RESPACKは資源を読取っている間、もしユーザのプロファイルにカストマイズされたバージョンがあるとすればオリジナルの代りに資源のカストマイズしたバージョンを代替 せるよう選択しうる．また、ユーザプロファイル内の資源は、それらが資源ファイルにあたかも記憶されているように編集しうる．

20.7.1. RESptcustid()－カストマイゼーションＩＤをセットする

この機能は、ユーザファイルからカストマイズされた資源を検索する上でRESPACKにより使用されるカストマイゼーションＩＤをセットする．このカストマイゼーションＩＤを介してカストマイズされた資源のみが別のRESPACK呼び出しにより戻される．ＩＤは現在のプログラムの実行の残りの部分を通して使用される．もしこの呼び出しが行われないとすれば、カストマイズされた資源は戻されることはない．

20.7.2. RESupedit()－プロファイルで資源を編集する

(79) RESPACKはユーザプロファイルに記憶された資源が、あたかも資源ファイルに記憶されていたごとくに編集できるようにする．この機能は、実際に使用されているファイルのタイプとは関係なくRESPACKルーチンを全体的に通して資源編集が実施できるようにする．

この機能はユーザプロファイルに関する情報を取得し、RESEFIDを必要とする他のRESPACK機能に対する引き数として使用可能のRESEFIDを戻す．全ての修正はプロフィルファイルに適用される．

プロファイルを編集するためにRESPACK機能のサブセットのみが使用できる．これらの機能とは、RESfcommit()、RESrdcur()、RESgtcur()、REScreate()、RESwrite()、RESrewrite()、RESptinfo()、およびRESdelete()である．その他の全ての機能は、プロファイルを修正するよう呼び出されるとエラーコードRESBADFIDを戻す．

前述の説明は本発明の特定実施例に限定してきた．さらに別の利点や修正が当該技術分野の専門家には明らかである．したがって、本発明は本明

細書に示し、かつ記述した特定の詳細、代表的装置並びに例示には限定されない．むしろ特許請求の範囲は本発明の真正な精神や範囲に入る前述のような変更や修正を網羅することである．

4．〔図面の簡単な説明〕

第１Ａ図、第１Ｂ図および第１Ｃ図は本発明を組み入れうる情報処理システムの線図、

第２図は本発明のシステムデータベースの線図、

第３図は本発明のオブジェクトマネージャの線図、

第４図は本発明のオブジェクトプロトタイプテーブルの線図、

第５図は本発明のオブジェクトカタログの線図、

第６図は本発明のオブジェクトテーブルの線図、

第７図は本発明のリンクテーブルの線図、

第８図は本発明のリンクに含まれる各種のデータ構造の線図、

第９図は本発明のマッチメーカを含む、本発明のデータ交換の線図、および

第10Ａ図と第10Ｂ図とは本発明の資源および資源エデイタの線図である．

代理人　弁理士　湯　浅　恭　三

（外４名）

(80)

図面の浄書（内容に変更なし）

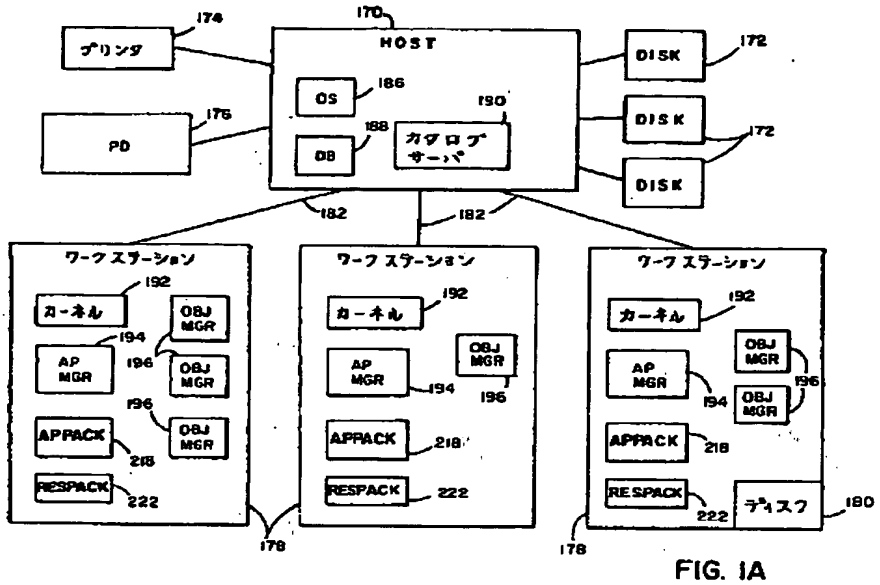


FIG. 1A

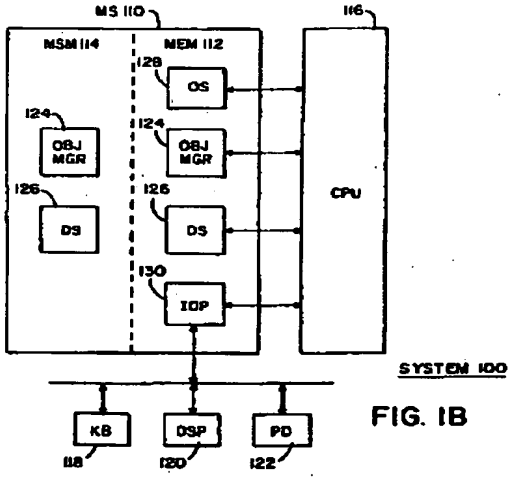


FIG. 1B

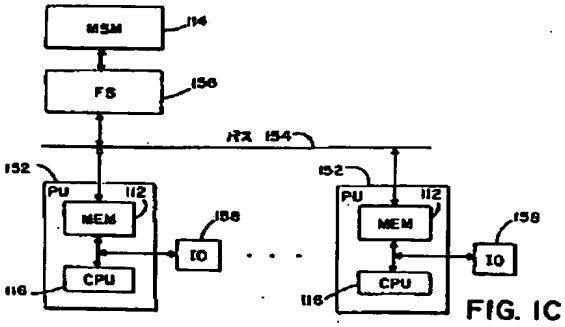


FIG. 1C

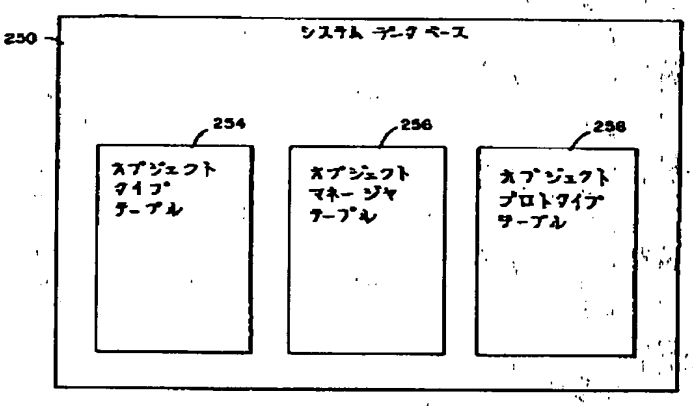


FIG. 2

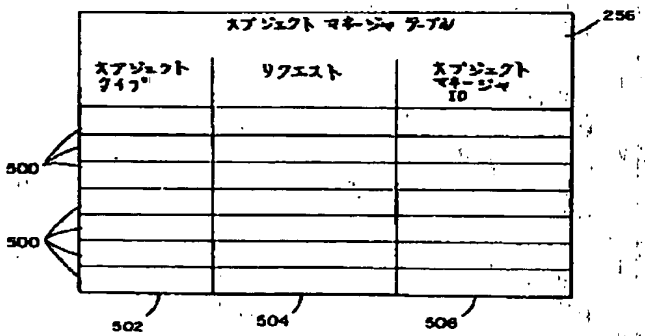


FIG. 3

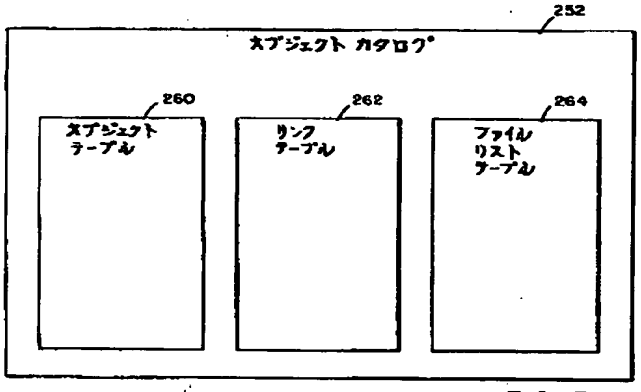


FIG. 5

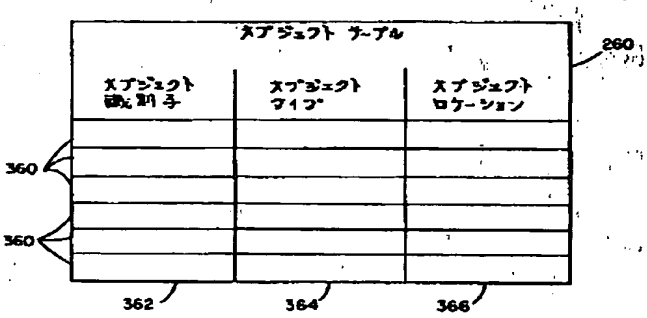


FIG. 6

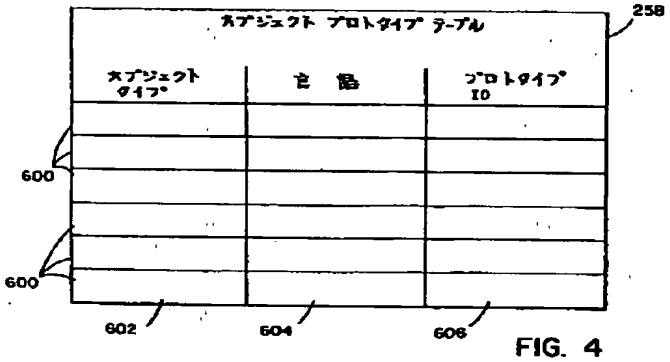


FIG. 4

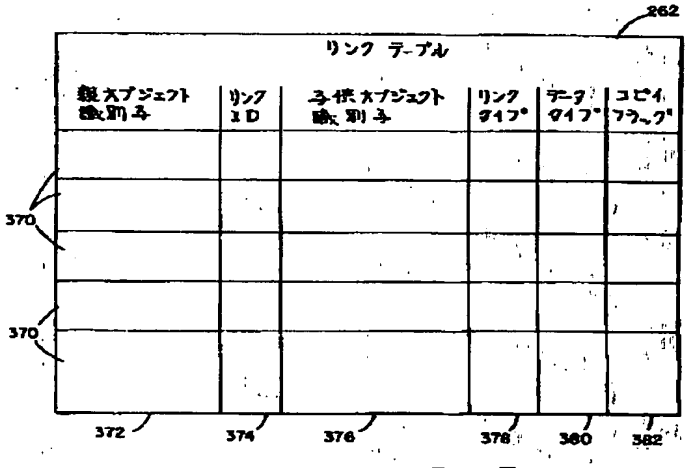


FIG. 7

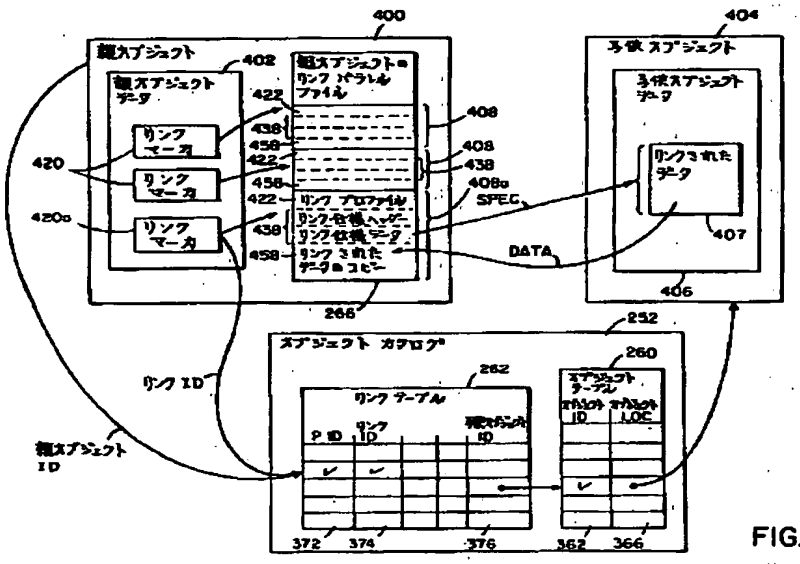


FIG. 8

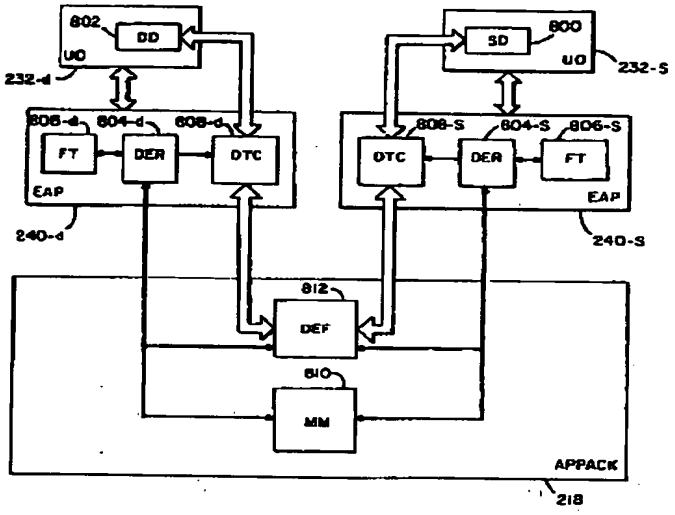


FIG. 9

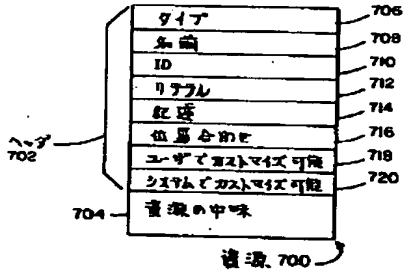


FIG. 10A

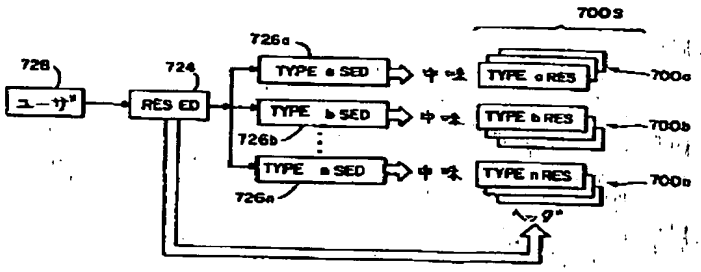


FIG. 10B

第1頁の続き

⑦発明者	キャロライン・イー・サプリナント	アメリカ合衆国マサチューセッツ州01826，ドラカット，コロニアル・ドライブ 28
⑦発明者	ローラ・オー・スターン	アメリカ合衆国マサチューセッツ州01801，ウオバーン，マンロー・アベニュー 26
⑦発明者	リーフオン・シ・フアム	アメリカ合衆国マサチューセッツ州01824，チエルムスフオード，グリーンバレー・ドライブ 9

手 続 補 正 書

昭和63年11月14日

特許庁長官　吉 田 文 毅　殿

1. 事件の表示

昭和63年特許願第207998号

2. 発明の名称

データ処理システム

3. 補正をする者
事件との関係　特許出願人
住 所
名 称　ウォング・ラボラトリーズ・インコーポレーテッド

4. 代 理 人
住 所　東京都千代田区大手町二丁目2番1号
新大手町ビル 206区
電 話 270－6641～6
氏 名 (2770) 弁理士 湯 浅 恭

5. 補正の対象
タイプ印書により浄書した明細書
適正な図面

6. 補正の内容
別紙の通り（尚、明細書及び図面の内容には変更なし）

方式審査 補本

（別紙）

(1)〔特許請求の範囲〕を次の通りに補正する。

『1．データがタイプされたオブジェクトとして表示されるデータ処理システムであって、

（A） 少なくとも1タイプのオブジェクトに対して少なくとも1オペレーションを各々が実行するようにされた複数のオブジェクトマネージャと、

（B）

(a) リクエストしているオブジェクトマネージャからオブジェクトの識別とオペレーションの識別とを受け取り、

(b) 前記の識別されたオペレーションを、前記の識別されたオブジェクトのタイプのオブジェクトに対して実行できるオブジェクトマネージャ、を識別し、および

(c) 前記の識別されたオブジェクトマネージャを呼び出し、かつ前記の識別されたオブジェクトに対し、識別されたオペレーションを実行するようにとのリクエストを識別されたオブジェクトマネージャに連絡するマネジメント手段と、

手 続 補 正 書

昭和63年11月14日

特許庁長官　吉 田 文 毅　殿

1. 事件の表示
昭和63年特許願第 207998 号

2. 発明の名称
データ処理システム

3. 補正をする者
事件との関係　特許出願人
住 所
名 称　ウォング・ラボラトリーズ・インコーポレーテッド

4. 代 理 人
住 所　東京都千代田区大手町二丁目2番1号
新大手町ビル206号室（電話 270－6641～6）
氏 名　(2770) 弁理士 湯 浅 恭

5. 補正の対象
明細書の〔特許請求の範囲〕の欄

6. 補正の内容
別紙の通り

方式審査 補本

を含むことを特徴とするデータ処理システム。

2． 他のオブジェクトへのリンクを有するタイプされたオブジェクトを処理するデータ処理システムであって、

（A） 各々が少なくとも1タイプのオブジェクトに対して少なくとも1オペレーションを実行するようにされた複数のオブジェクトマネージャと、

（B） オブジェクト間のリンクを識別し、かつ各リンクと関連する前記オブジェクトを識別する情報を記憶する記憶手段と、

（C） 前記の記憶されたリンク情報をアクセスし、かつ前記の複数のオブジェクトマネージャの全てに対するリンク情報を供給するために使用可能なアクセス手段と、

（D）

(a) リクエストしているオブジェクトマネージャからリンクの識別およびオペレーションの識別を受取り、

(b) 前記の識別されたリンクによって指示された前記オブジェクトに対して前記の識別され

たオペレーションを実行できるオブジェクトマネージャ、を識別し、

(c) 前記の識別されたオブジェクトマネージャを呼び出し、

(d) 前記の識別されたオブジェクトに対して、前記の識別されたオペレーションを実行するようにとのリクエストを、前記の識別されたオブジェクトマネージャに連絡するマネージメント手段と、

を含むことを特徴とするデータ処理システム。

3. 各々が少なくとも1タイプのオブジェクトに対して少なくとも1オペレーションを実行するようにされた複数のオブジェクトマネージャを有するオブジェクトベースのデータ処理システムにおいて、アプリケーションマネージャが、

(A) リクエストしているオブジェクトマネージャからオブジェクトの識別とオペレーションの識別とを受け取る手段と、

(B) 前記の識別されたオブジェクトのタイプのオブジェクトに、前記の識別されたオペレー

ションが定義されており、かつ前記の複数のオブジェクトマネージャが、各タイプのオブジェクトに対して前記の対応する省略時オペレーションを実行できる少なくとも1個のオブジェクトマネージャを、含むことを特徴とするデータ処理システム。

7. 請求項2に記載のシステムにおいて、各タイプのオブジェクトに対して省略時オペレーションが定義されており、かつ前記の複数のオブジェクトマネージャが、各タイプのオブジェクトに対して前記の対応する省略時オペレーションを実行できる少なくとも1個のオブジェクトマネージャを、含むことを特徴とするデータ処理システム。

. 請求項1に記載のシステムにおいて

(i) 前記マネジメント手段がマネージャプロセスからなり、

(ii) 前記のマネジメント手段による受け取りが、前記のマネージャプロセスによる、リクエストしているオブジェクトマネージャからのプロセス間通信の受け取りを含み、および

(84) ションを実行できるオブジェクトマネージャ、を識別する手段と、

(C) 前記の識別されたオブジェクトマネージャを呼び出し、前記の識別されたオブジェクトに対し、前記の識別されたオペレーションを実行するようにとのリクエスト、を前記の識別されたオブジェクトマネージャに連絡する手段とを含むことを特徴とするデータ処理システム。

4. 請求項1に記載のシステムにおいて、前記の複数のオブジェクトマネージャが、各タイプのオブジェクトに対して1組の標準オペレーションの各々を実行できるオブジェクトマネージャを含むことを特徴とするデータ処理システム。

5. 請求項2に記載のシステムにおいて、前記の複数のオブジェクトマネージャが、各タイプのオブジェクトに対して一組の標準オペレーションの各々を実行できるオブジェクトマネージャを、含むことを特徴とするデータ処理システム。

6. 請求項1に記載のシステムにおいて、各タイプのオブジェクトに対して省略時のオペレー

(iii) 前記のマネジメント手段による呼び出しが結果的に前記マネージャプロセスにオブジェクトマネージャプロセスを生成させるようにすることを特徴とするデータ処理システム。

9. 請求項1に記載のシステムにおいて、前記マネジメント手段が、オブジェクトをオブジェクトタイプに関連させるデータベースをアクセスする手段を、含むことを特徴とするデータ処理システム。

10. 請求項1に記載のシステムにおいて、前記マネジメント手段が、オブジェクトタイプをオブジェクトマネージャに関連させるデータベースをアクセスする手段を、含むことを特徴とするデータ処理システム。

11. 請求項1に記載のシステムにおいて、前記のリクエストしているオブジェクトマネージャにより提供されたオブジェクトの識別がリンク識別子からなることを特徴とするデータ処理システム。

12. タイアされたオブジェクトとしてデータが表示されるデータ処理システムにおいて、

(A) 少なくとも1タイアのオブジェクトに対して少なくとも1オペレーションを各々実行するようにされた複数のオブジェクトマネージャと、

(B)(a) オブジェクトをオブジェクトタイアと関連させ、

(b) リンク識別子を特定のオブジェクトと関連させ、

および(c) オブジェクトタイアをオブジェクトマネージャと関連させる

システムデータベースと、

(C)(a) リクエストしているオブジェクトマネージャからリンク識別子とオペレーションの識別とを受け取り、

(b) システムデータベースにアクセスして、前記リンク識別子が参照しているオブジェクトを識別し、

(c) 前記システムデータベースにアクセスして、識別されたオブジェクトのタイアを検出し、

(d) 前記システムデータベースにアクセスして、前記の識別されたオブジェクトのタイアの

含むことを特徴とするデータ処理システム。

14. 請求項13に記載のシステムにおいて、前記記憶手段が、前記リンクがデータリンクであるか否かの指示を各リンクに対して含むことを特徴とするデータ処理システム。

15. 請求項13に記載のシステムにおいて、リンクの中の少なくともあるリンクは子供オブジェクトからのデータを親のオブジェクトにリンク結合させ、前記の記憶手段が各リンクに対して、

(i) 前記の子供オブジェクトからリンクされたデータが修正されるとき、

(ii) 前記の親オブジェクトが開放されたとき、あるいは

(iii) 特別の更新リクエストがなされたときのみ、

リンクが更新されるべきことを各更新状態が指示できる更新状態の値を記憶することを

特徴とするデータ処理システム。

16. リンクされたオブジェクトを含むデータ処理システムにおいて、

(85) オブジェクトに対し、前記の識別されたオペレーションを、実行できるオブジェクトマネージャを識別し、および

(c) 前記の識別されたオブジェクトマネージャを呼び出し、前記の識別されたオブジェクトに対し、前記の識別されたオペレーションを実行するようにとのリクエストを前記の識別されたオブジェクトマネージャに連絡する

マネジメント手段、

とを含むことを特徴とするデータ処理システム。

18. リンクされたオブジェクトを含むデータ処理システムであって、

(A) 複数のオブジェクトマネージャと、

(B) 複数のオブジェクトと、

(C) 各リンクと関連した前記のオブジェクトを識別する情報を記憶する記憶手段と、

(D) 前記の記憶されたリンク情報にアクセスする手段であり、前記の複数のオブジェクトマネージャの全てに対してリンク情報を供給しうるアクセス手段と、を

(A) 複数のオブジェクトマネージャと、

(B) リンクマーカを含む複数の親オブジェクトと、

(C) リンクマーカを含むオブジェクトからは分離しており、子供のオブジェクトをリンクに関係づける情報を記憶する手段と、

(D)(i) リクエストが、前記リンクを識別することにより、リンクされたオブジェクトを識別する場合に、リンクされたオブジェクトに対してオペレーションを実行すべきとのリクエストを受け取り、および

(ii) 前記の記憶されたリンク情報にアクセスし、どのオブジェクトが前記のリクエストで識別された前記リンクの子供であるか検出するために、前記のオブジェクトマネージャの各々にとって利用可能のリンクマネジメント手段

と、を含むことを特徴とするデータ処理システム。

17. 請求項16に記載のシステムにおいて、前記リンクマネジメント手段が、受け取られたリクエストに応答して、前記の識別されたリンク結合す

みオブジェクトに前記のリクエストされたオペレーションを実行するようオブジェクトマネージャを呼び出すことを特徴とするデータ処理システム。

18. 請求項17に記載のシステムにおいて、前記リンクマネジメント手段が応答するリクエストは、リンク更新リクエストであることを特徴とするデータ処理システム。

19. 請求項16に記載のシステムにおいて、各リンクマーカは、該マーカが配置されているオブジェクト内では一意であるリンク識別子を含み、かつリンクは、前記該オブジェクトと、対応するリンクマーカの前記リンク識別子と、を識別することにより識別されることを特徴とするデータ処理システム。

20. 請求項19に記載のシステムにおいて、リンクされたデータのコピイを記憶する、リンクされたデータのコピイの記憶手段をさらに含むことを特徴とするデータ処理システム。

21. 請求項20に記載のシステムにおいて、前記のリンクされたデータのコピイの記憶手段が、別

突き合わせデータ交換の役割を有するプロセスの間の通信をアレンジする手段をさらに含むことを特徴とするマルチ処理システム。

24. 請求項22に記載のシステムにおいて、データ交換の役割が、コピイ、移動、共用および位置づけオペレーションを含むことを特徴とするマルチ処理システム。

25. ユーザの入力のあるものが第１のプロセスに導かれ、ユーザの入力の他のものが第２のプロセスに導かれるマルチ処理システムにおいて、マッチメーカのサポートを得て第１のプロセスと第２のプロセスとの間でユーザ主導のデータ交換を行う方法が、

（A） 前記第１のプロセスがデータ交換オペレーションにおいて果すべき役割を規定した入力を、前記第１のプロセスに前記ユーザが供給するステップと、

（B） 前記第１のプロセスに対して規定された役割の指示を前記マッチメーカに前記第１のプロセスが通知するステップと、

(86) のオブジェクトへのデータリンクを有する各オブジェクトに対応する個別のファイルにおいて記憶することを　徴とするデータ処理システム。

22. ユーザの入力のあるものが第１のプロセスに導かれ、ユーザの入力の他のものが第２のプロセスに導かれるマルチ処理システムにおいて、マッチメーカが、

（A） 複数のプロセスのいずれかから、指示しているプロセスの果すデータ交換の役割の指示を受け取る手段と、

（B） 受け取られた役割の指示を記憶する手段と、

（C） 各々の受け取られた役割の指示が、記憶された役割の指示と適合するか否かを検出する突き合わせ手段と、

（D） そのプロセスから受け取った役割の指示が別のプロセスから受け取られた役割の指示と適合した旨をプロセスに通知する手段

とを含むことを特徴とするマルチ処理システム。

23. 請求項22に記載のシステムにおいて、その

（C） 前記第２のプロセスがデータ交換オペレーションにおいて果すべき役割を指示する入力を前記第２のプロセスにユーザが供給するステップと、

（D） 前記第２のプロセスに対して規定された前記役割の指示を、前記マッチメーカに前記第２のプロセスが通知するステップと、

（E） 前記第２のプロセスに対して規定された役割が前記第１のプロセスに対して規定された役割と適合することを前記マッチメーカが検出するステップと、

（F） 前記マッチメーカが前記第１と第２のプロセスの間の通信をアレンジするステップと、

を含むことを特徴とするデータ交換の方法。

26. 請求項25に記載の方法において、前記の通信をアレンジするステップが各プロセスに適合したことの検出を通知し、かつ各プロセスに、該プロセスが他のプロセスとの通信において用いうるコードを供給することを特徴とするデータ交換の方法。

27. 請求項25に記載の方法に いて、データ交換の役割が、コピイ、移動、共用および位置づけオペレーションを含むことを特徴とするデータ交換の方法。』 (87)

以 上